MSX BASIC

MET VPOKE EN SPRITE TOEPASSINGEN

WIE ZEGT DAT U DAT NIET KUNT PROGRAMMEREN

J.G. Ottenhoff

MSX Basic met vpoke en sprite toepassingen

Wie zegt, dat u niet kunt programmeren?

J.G. Ottenhoff

uitgeverij STARK-TEXEL
postbus 302 - 1794 ZG Oosterend tel. 02223 - 661

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Ottenhof, J.G.

MSX Basic vpoke en sprite toepassingen : wie zegt, dat u niet kunt programmeren: / J.G. Ottenhof. – Oosterend : StarkTexel
ISBN 90 6398 372 7
SISO 365.3 UDC 681.3.06+800.92 MSX-basic NUGI 852
Trefw.: MSX-basic (programmeertaal) / programmeren (computer).

1e druk 1986
ISBN 90 6398 372 7

Inhoudsopgave

pagina

hoofdstuk 1
Het hoe en waarom

Misschien is het u ook overkomen. Jarenlang computeren zonder MSX maar met plezier. En met de voldoening over redelijk geslaagde programma's. Tussendoor wél eens de frustrerende gedachte, dat het allemaal uitsluitend op jouw fabrikaat wil draaien. Soms scheelt het maar een enkele komma met de Basic-versie van andere produkten. Maar toch. Daarbij komt dan, dat de uitrusting nooit kompleet is. Er moet altijd van alles worden bijgekocht om net iets meer te kunnen doen met de machine. Een commercieel beleid van ondernemingen, die dit tenslotte zelf lelijk is opgebroken. Produktie gestaakt en miljoenenverliezen proberen weg te werken. Evengoed blijft het plezier van de hobby. Veel geld is er aan besteed en niet te vergeten zijn die programma's, gedurende enige jaren bijeen gecomputerd. Wat moet je er mee? Je dacht nog wel aardig thuis te zijn in dat rijk der variabelen, funkties en kommando's. Ja, maar dan wél voornamelijk in dat eigen Basicwereldje van die fabrikant. Een beetje ontmoedigend is zo'n gevolgtrekking Dan maar iets anders verzinnen om vrije tijd te vullen? Voetballen of pingpongen wellicht . . .
Een goede vraag kan zijn voor wie het de grootste stap is om een homecomputer aan te schaffen. Voor wie net begint of voor wie al tijden een andere in gebruik heeft. Je staat als een kat in een vreemd pakhuis aan te kijken tegen Basic-instrukties, waarvan je het bestaan niet eens vermoed had. Ook met de nodige argwaan en afwijzing. Je moet min of meer terug naar AF en als een nieuweling onder computeraars van voren af aan beginnen. Zij het wél met die ene, belangrijke zekerheid, dat je met de nieuwe MSX-machine standaard in huis hebt gehaald. Totaal geen problemen meer met komma's.
De gedachte dat jouw programma voortaan op vele computers van diverse fabrikanten kan draaien, is dan voldoende stimulans om je in dat nieuwe van MSX te verdiepen. Weliswaar blijkt het dan allemaal niet zó volkomen te zijn, als wel eens wordt voorgesteld, maar in veel opzichten minstens 300% beter dan wat je gewend was. Aldus niet geheel zonder weemoed de oude machine netjes terug gezet in de verpakking. Met wat er zo allemaal bij was aangeschaft. Maar dan die programma's? Daar zijn er bij, die je beslist volgens MSX-Basic opnieuw wilt programmeren. Omdat je ze nodig hebt of gewoon omdat je ze te

aardig vindt om ze te laten verstoffen. Nou ja, wij beoordelen nu eenmaal sterk subjectief. Onze programma's zijn het helemaal. Flauwekul natuurlijk. Het kan altijd beter. We blijven amateurs.
Ervaring met een home-computer geeft zeker enige voorsprong, wanneer je je waagt aan MSX. En dát kunnen beginners niet zeggen. Aanvankelijk evenwel zit je toch in hetzelfde schuitje. Maar de opnieuwbeginner mist iets.
Iets heel essentieels dat hij voordien wél ter beschikking had.
Alom valt de nadruk op de grafische mogelijkheden van screen2 en op de sprites. Spectaculair is het wel, maar sprites waren je toch al niet onbekend. Weldra ontdek je trouwens, dat screen2 niet bepaald het meest aangewezen scherm is om mee te werken. Of je moet dat grafische per sé nodig hebben. Dit scherm kent beperkingen, zoals ook door de tekst-file, die in algmeenheid doet kiezen voor screen1. Het verreweg plezierigste scherm. Hoe krijg ik echter daar bijvoorbeeld kleurvlakken en figuratie op? Zelfs met Draw is het niet niks om op screen2 een wat ingewikkelder figuur neer te zetten. Het werd snel duidelijk, dat van een herschrijven van oude programma's weinig terecht zou komen. Allereerst zou informatie moeten worden verzameld. Ongenoemd in handleidingen en vaak ook in MSX-boeken voor beginners. De basis van MSX-Basic. Naast het doorsnuffelen van boeken en MSX-bladen, het nalopen van andermans programma's in die bladen. En diepteonderzoek in de adressen van vooral VRAM in de hoop op positieve resultaten. Nu wil ik tóch altijd al graag weten hoe iets in elkaar zit en funktioneert. Dat kon dus op die manier.
Waarom het ging, dat was eigenlijk nergens voldoende aangegeven. In het gunstigste geval zijdelings in dan nog vooral boeken over machinekode. Kennelijk werd gezocht naar de stiefkinderen van MSX-Basic. Er is in elk geval weinig zinnigs mee gedaan in programma's, waarvan de listings in de overigens uitstekende MSX-magazines worden afgedrukt. Zelfs in de engelstalige bladen waren de gezochte instrukties nauwelijks terug te vinden.
Heel merkwaardig. Ook de aanduiding Sprite-editor voor programma's waarmee eigen figuren te ontwerpen zijn, versterkte de gedachte, dat met MSX kennelijk uitsluitend sprites als figuranten in een programma te gebruiken zijn. Of met Draw natuurlijk. Begin er maar eens aan bij een ingewikkelde figuur.
Het vreet trouwens bytes. Bovendien wil je helemaal niet met screen2 werken.
Waarom het nu eigenlijk gaat? Heel simpel. Zoals dat met de basis der dingen toch doorgaans het geval is. Hoe krijgt een mens kleurvlakken of balken op screen1 al dan niet met tekst met dezelfde achtergrondkleur? Welke toetsen moet iemand van goede wil aanslaan om één van die MSX-karakters op het scherm te krijgen, die niet op de toetsen zijn

vermeld? Hoe verander je die karakters in zelf ontworpen figuren. Neem nu dat schaakbord-programma, dat als laatste onderdeel van dit boek is gemeend. Acht schaakstukken op één rij? Met sprites kan dat niet. En hoe verf ik een en ander in de beschikbare kleuren? IF . . . THEN! Dan nog de schermlezing. Hoe kom ik te weten waar wat staat? Dat is dikwijls nodig in een programma. Nu, voor deze basis-dingen bezat de oude computer heel gebruiksvriendelijke instrukties ondanks alle beperkingen.
Gewoon CALL voor kleur, karakterwijziging, printen van kleurbalken en vlakken en de schermlezing. Dat is er in MSX niet bij. Dat hoeft trouwens ook niet. Zolang iemand echter niet vertrouwd is met dat arsenaal van VRAM en het gebruik van VPOKE en VPEEK, blijft dit de basis aan Basic onthouden.
De MSX-computer is een geweldig stuk gereedschap. Weliswaar ook niets meer dan dat, doch het onderscheidt zich wel heel gunstig van andere home-computers, daarom zou het jammer zijn als de machine tot een kastcomputer wordt gedegradeerd. Of in het gunstigste geval alleen benut wordt voor gekochte spelletjes. Uit alle publikaties rondom MSX ontstaat de indruk, dat Nederlanders geniale computeraars zijn. Dat is echter niets meer dan het bekende topje van de ijsberg. De echte keien vormen een betrekkelijk bescheiden groep. En hún listings komen in de bladen en zij behoren tot de publicisten. Zowel voor de beginner als voor wie MSX de vervanger is van de andere computer, kan één en ander nogal ontmoedigend zijn. Waar zijn we aan begonnen? Wij worden nooit keien. Nu ja, zeg nooit nooit natuurlijk. Maar het zal bepaald geen uitzondering zijn, dat iemand die zo vol goede moed begon, intussen z'n MSX aan de wilgen heeft gehangen. Want wat hij ermee wilde doen, dat schijnt niet te kunnen. Maar het kan wel. Heel goed zelfs. U kunt het ook. Computeren is als een denksport. Enige creativiteit en logika heeft het nodig. Als we maar weten hoe.
Wat dan zoal bijeenverzameld is aan basis-informatie, dat kun je natuurlijk lekker voor jezelf houden. Het leek me echter een goede zaak om het aan u door te geven. Want met deze basis van BASIC kunt u werkelijk heel wat kanten uit. Dat kon ik tenslotte ook. Na enkele maanden. Zelfs met dat schaakbord. En wie een MSX bezit, die moet gewoon zelf aan het programmeren slaan. Laat het dan maar een spelletje zijn. Al te veel funktionele programma's zal de doorsnee computeraar nu ook weer niet nodig hebben. Het belangrijkste is de eigen voldoening. En de toepassing van de Basic-instrukties met alle opgedane ervaring vandien. Er is altijd wél een zekere moed voor nodig. Het vergaat de programmeur als de schrijver voor een leeg blad papier. Hoe beginnen en wat moet er allemaal in worden ondergebracht? Hier is ook de grondigste kennis van Basic ontoereikend. Iemand die zelden een brief schrijft, verontschuldigt zich ermee niet te weten wát te

moeten schrijven. Maar schrijven kan hij heel goed. Zo ook de computeraar. En al te veel begeleiding daarbij ontvangt hij niet. Het zal er de oorzaak van zijn, dat er ook in MSX-land vooral veel wordt geëxperimenteerd. Bijvoorbeeld om HOOKS te ontdekken of om andere truukjes uit te halen. Kan op zichzelf heel boeiend zijn. Maar een eigen, goed werkend programma is hoe dan ook de bekroning van de hobby. Helemaal niet omdat het een kick zou geven de computer de baas te zijn. Dat is flauwekul.
Het is alleen úw logika welke kan triomferen.
Het helpt soms, wanneer je je zelf wijsmaakt, dat je een programma in opdracht moet maken. Dat er, ik weet niet wat, van afhangt om het zo goed mogelijk te doen. Zo alsof je ermee voor de dag moet kunnen komen tegenover 14 miljoen Nederlanders en de nodige minderheden. En dat die allemaal heel kritisch jouw programma zullen uitkammen. Op zoek naar overbodigheden en slordigheden. Dat kan helpen. Ook de computeraar groeit tegen de verdrukking in.
Kortom, de MSX-computer biedt zoveel mogelijkheden, dat het zondermeer jammer zou zijn, indien deze niet zo uitputtend mogelijk ook in uw programma gebruikt zouden worden. Daarom is de voorzichtige verwachting, dat deze regels u daaarbij een duwtje in de goede richting zullen geven. Eigenlijk worden er maar enkele zaken behandeld. Echter wél juist die stiefkindjes van MSX. Al te ingewikkeld willen we het al evenmin doen. Daarom is het één en ander aan de hand van een tiental programma's onderstreept. Dat zegt meer dan duizend woorden. Tenslotte hebben we daarvoor onze computer. En standaard! U zult er geen komma's voor hoeven te vervangen. Alle listings van deze programma's treft u aan in APPENDIX-VI, waaronder enkele korte listings. De meeste echter zijn kombinaties en het ontbreekt niet aan Data-regels. Het is altijd een hele klus om listings foutloos in te tikken. Een vergissing is snel gemaakt. U kunt er alleen aan ontkomen, indien u bij de uitgever een cassette of disk bestelt. Echter, vooral als u nog niet al te veel ervaring hebt, dan is het toch een goede zaak deze listings zelf in te tikken. Al vraagt het wat van uw tijd en zorgvuldigheid. U raakt er dan nauwer bij betrokken. En ik wil u voorstellen eerst een programma in te tikken vanaf de listing zodra we deze nodig zullen hebben. Al zijn het primair hulpprogramma's om de computer te laten demonstreren waarom het gaat, ze zijn toch ook iets meer dan dat. Er is zo het één en ander in verwerkt en toegepast, dat u op ideetjes zou kunnen brengen. Er valt mee te knutselen. Van tenminste twee programma's zult u blijvend plezier kunnen hebben. Het zou jammer van al uw ingetypte moeite zijn, als het alleen om wegwerpprogramma's ging. Het is de bedoeling, dat u blijvend plezier zult hebben aan de aanschaf van dit boek. Aldus; zodra we in de loop van dit verhaal aan een programma toe zijn, dan zal ik u stimuleren om de betreffende listing in te tik-

ken.

Aangaande deze programma's nog een paar woorden. Ik zal beslist de allerlaatste zijn, die ook maar iets wil pretenderen inzake zijn programma's. Daaraan heb ik niet de minste behoefte. Het is altijd voor verbetering vatbaar. Behalve dat mijn MSX-verleden nogal recent is, zijn die programma's algemeen wat wijdlopend opgezet terwille van de duidelijkheid.
Dat is iets anders dan efficiënt programmeren. Vroeger stond er in de leesboekjes van de lagere school: eerst duidelijk, dan snel. Het zal ook een goed startpunt zijn voor de computeraar. Moeilijk kan altijd nog. We moeten het zelf wél blijven snappen. Ook als we na tijden weer een eigen programma-listing onder ogen krijgen. Daarom zijn het min of meer programma's gemeend om er wat uit te halen als om zelf mee te knutselen. Het gaat er uitsluitend om, dat één en ander een aanzet kan geven om tot eigen programma's te komen. Want indien u eenmaal met deze basis van Basic vertrouwd bent en u de overige instrukties goed weet te gebruiken, dan zou er niets meer mis mogen gaan. Dan schrijft u weldra uw eigen programma. En wie weet, waartoe dat nog zal kunnen leiden. Nu wij deze afspraak aangaande het intikken van de listings hebben gemaakt, is het langzamerhand tijd geworden om maar van start te gaan. Aldus RUN te geven!

hoofdstuk 2
De onderwereld van VRAM

U zou weinig aan dit boek hebben, als het niet in voor u te begrijpen karakters zou zijn geschreven. Ofschoon dit voor de computer net zo is, wordt daarbij geen rekening gehouden met wat wij meteen kunnen begrijpen. Voor de interne operaties bedient men zich er van een eigen taaltje. Alleen voor ons wordt een resultaat in te begrijpen taal op het scherm gezet.
Evenzeer maken we gebruik van letters, cijfers, leestekens en symbolen bij het programmeren. Maar dat is een hulpmiddel waarvoor men intern de neus ophaalt. Dat schakelcircuit geeft de voorkeur aan binaire babbels. Het is tenslotte niets anders dan een ingenieuze samenstelling van elektronische schakelingen. En een schakelaar zet je aan of uit. In dat binaire staat de 1 dan ook voor "aan" en de 0 voor "uit". Een onderscheid met het vertrouwde lichtknopje is alleen, dat zo'n MSX-schakelaar op 256 verschillende manieren geschakeld kan worden. Maar daar moet dan wél alles mee gedaan worden bij de uitvoering van onze opdrachten. Dat lukt. Er zitten trouwens nogal wat van deze schakeleenheden in onze computer. U zou het aan lichtknopjes niet in huis willen hebben. Feitelijk is zo'n eenheid wat een adres wordt genoemd. Al die adressen hebben wel een eigen nummer, zoals dit sinds Napoleon algemeen gebruikelijk is, maar wie er woont, dat kunnen we goeddeels zelf bepalen. Alleen de stad ROM heeft vaste inwoners. Daar verhuist niemand en evenmin laat zich er iemand door ons van zijn adres verjagen. Ze zijn er wat je noemt honkvast. Dit geldt overigens deels ook voor de buitenwijk van ROM; het zo genoemde systeemgebied helemaal boven in het adressenbestand van RAM. Daar kán, hoe tijdelijk ook, zo het één en ander veranderd worden.
Dat zal echter vooral een klus zijn voor de keien. Waartoe wij ons (nog) niet mogen rekenen. Het totaal van het vrij programmeerbare adressenbestand noemt men dus het geheugen. Of minder vermenselijkt: het aantal nog te benutten schakelaars. Door dit schakelen slaan we een signaal op, dat in kombinatie met soortgenoten de diverse processors aan het werk zet. De CPU, als de Z-80 microprocessor in de MSX, is zoiets als het bestuurlijke centrum van ROM. De afdelingen van dit centrum, als de registers van deze processor, hebben ook in de computer de handen vol om allle kronkels van de programmeur te verwerken.

De beleidskommissie houdt er een serie trefwoorden bij de hand, om bij een negatieve konklusie op het scherm te kunnen projekteren: ERROR! Verzoek afgewezen en nog maar eens proberen. En als het even kan beter. Het funktioneert allemaal heel soepel en vooral heel snel. Bijna nog voordat wij de fout hebben gemaakt, is deze al vastgesteld door die CPU. De onverbiddelijke rekenmeester in onze computer. We hebben Basic om niet al te intensief betrokken te worden bij al dat schakelen. In de MSX wordt het allemaal omgezet in informatie, waarmee men in ROM echt iets kan aanvangen. Dat is een prettige zaak. Het is zoiets als autorijden met een automatische versnellingsbak.
De sportieve rijder schakelt liever zelf. Willen we die basis van Basic leren kennen, dan moeten we ook iets van dat schakelen weten. Niet al te veel overigens. Behalve adres wordt zo'n schakeleenheid ook aangeduid als byte. Omdat de bewoner van het adres evenzeer met byte wordt aangeduid, wordt zo onderscheid gemaakt tussen adres en byte. Het is iets met 8 knopjes en die worden weer bits genoemd. Sinds de eerste computer is er een hele woordenschat bijgekomen. MSX biedt vier verschillende kodes om een schakeling te programmeren. De decimale, hexadecimale, octale en binaire kode. De eerste is ons het meest vertrouwd en de laatste geniet dus de voorkeur in het interne gebeuren van de computer. Ze kunnen alle vier worden benut en in veel gevallen desgewenst afwisselen. Voor ons zal de herkenbaarheid en de beknoptheid bepalend zijn bij de keuze. Wel wordt soms de binaire kode aanbevolen, omdat deze door de nullen en enen de meest realistische weergave geeft van een schakeling. Maar een beetje binaire kode bestaat al gauw uit 8 getallen. Hoe lang zouden DATA-regels dan wel niet worden? Daarom kunnen we het binaire beter laten voor wat het is voor de computer. Eerst als we ons met machinekode bezig gaan houden, zullen we er nog vroeg genoeg mee te stellen krijgen.
De decimale kode is wél de ons meest vertrouwde. Echter lang niet altijd direkt naar de betekenis herkenbaar en evenmin de beknopste. Het is gewoon lastig om te onthouden waarvoor bijvoorbeeld decimaal 126 staat. Daarvoor zal steeds een tabel moeten worden geraadpleegd. Programmeren kost toch al zoveel tijd. In beginsel is de decimale kode nog het beste te gebruiken in lussen voor adressen of om aan een variabele zo'n waarde toe te kennen. De kode waarmee wij het meeste te maken zullen krijgen is de hexadecimale. Kortweg als hex geschreven. Het bestaat uit slechts twee getallen. Dat doet wat vreemd aan. Het bedient zich immers ook van letters: 0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F. Die vervangen echter niet bestaande enkelvoudige cijfers. Vandaar. De hex-kode, voorafgegaan door &H, is zowel de beknopste als de gemakkelijkst te herkennen kode.





Deze voorstelling van de byte mag het duidelijk maken. Voor hex is het in twee parten verdeeld. Er bestaan slechts 16 manieren om de bits in één van die partjes te schakelen ofwel meer grafisch gezien, om de hokjes ervan zwart te maken. Of geen enkel hokje natuurlijk. Dan blijft dat deel op "uit" staan. Deze 16 als boven bijeen gezet zijn rap onder de knie te krijgen. Dat moet. Een hex-kode bestaat altijd uit twee getallen. Al is het geen noodzaak om de eerste 0 te noteren, als dat althans een 0 zou wezen. We hebben het bovendien over de bewoner van een adres en niet over zijn adres. Een adresnummer loopt op tot 4 getallen. Het eerste getal van de hex-kode slaat op het eerste part, ofwel de eerste vier bits van de byte. En het tweede getal . . . u hebt het geraden! Stel dat we in part-1 het meest linkse hokje zwart maken. Niet rood. Het gaat hier niet over verkiezingen. Dan hebt u het verkeerde boek gekocht. Dit zwart maken levert de hex-kode 8 op. In part-2 kleuren we het meest rechtse hoekje terwille van de veelvormigheid. Dat is goed voor de hex-kode 1. Zo wordt in hex uitgedrukt deze schakeling weergegeven als &H81. Niet zo ingewikkeld. Wat oefening en ervaring brengt weldra het beeld van de geschakelde byte voor ogen, als we daarvan de hex-kode kennen.
Daarmee krijgen we te maken in het pakhuis van de video display processor, kortweg VDP genoemd. Het is de magazijnmeester van het VRAM-blok. Wat in dat pakhuis zit of komt te zitten bevat de informatie nodig om tekst en figuratie op het scherm te brengen. De innige samenwerking met de sound-processor (in die omgeving bekend als de PSG) draagt eventueel zorg voor ondersteunende deuntjes. Niet alleen is dit VRAM-blok een afzonderlijk bestand van adressen. De waardering die wij hebben op te brengen voor al dan niet geschakelde bytes daarin is een geheel andere dan voor hun familieleden in ROM. Want in VRAM vertegenwoordigen de bytes uitsluitend de informatie over bijvoorbeeld de karakters die we op het scherm willen kombineren tot enigzins leesbare teksten. In ROM moeten ze daar niets van hebben. Daar zijn ze strikt zakelijk ingesteld. Om ons terwille te zijn wordt met INs en OUTs vanuit ROM een beroep gedaan op het arsenaal van de VDP. De bytes in ROM zijn de werkpaarden.
Middels deze vaste schakelingen wordt allles verwerkt. Het moet frustrerend voor deze knollen zijn, dat zij tenslotte toch weer een beroep op het elitaire gezelschap in VRAM moeten doen, om een resultaat op het scherm te kunnen tonen. ROM is een stad vol werkslaven, gebonden aan huis en haard, die nooit eens op de voorgrond kunnen treden. Ja, als wij aan het knoeien zijn en er een foutmelding op het scherm gezet

moet worden. Tamelijk beroerde sociale omstandigheden zou je denken. Het nadeel van de VRAM-burgers is evenwel, dat zij altijd in het ongewisse blijven omtrent hun uiterlijke verschijning. Bij hun ontwaken, als wij de computer aanzetten, moet de A maar afwachten of hij niet in een Z veranderd zal worden. Wij hebben die macht. Als ware despoten. Net zo gemakkelijk kan het lachebekje, kodenummer 1, veranderd worden in een groen blokje, waarmee wij een vlak op het scherm willen PRINTen. Meer positief afwachtend zijn hier de bytes bedoeld voor de sprites. Hun ledigheid kan door ons worden opgeheven.
Laten we ons evenwel nog even bezighouden met die 256 schakelmogelijkheden van zo'n byte. In APENDIX-II zijn ze allemaal zowel in decimaal als in hex opgenomen. Het brengt ons bij het eerste programma. Dat wordt programma nr.1 BYTES. Nadat u het heeft ingetypt, kan ik er eerst wat over vertellen. Het heeft maar één onderdeel. Dat laat u die 256 schakelmogelijkheden met de bijbehorende kodes zien. Een goede gelegenheid om zelf te constateren, dat hiervan de hex-kode uit slechts twee getallen blijft bestaan. Het is goed om die kode mee te lezen. Het leert heel snel. De wijze van schakeling is bepalend voor het signaal, dat uiteindelijk in de microprocessor voor verwerking terecht zal komen. In ROM is het dus allemaal voorgeprogrammeerd. De schakelingen liggen vast. Dat de uitvoering van een routine in ROM desondanks te beïnvloeden is via de hooks, dat doet hieraan niets af. De clock-pulse-generator als de drijvende kracht achter de CPU tikt weliswaar liefst 3½ miljoen keer per sekonde, maar dat is iets anders dan de snelheid waarmee de processor alle informatie verwerkt. Daarbij hangt die snelheid uiteraard mede af van het soort verwerkingen. Bij machinekode wordt deze verwerkingssnelheid gemeten in klokcycli. Typen we TIME(0) met PRINT of ? ervoor, dan is vast te stellen hoe rap dit uurwerkje loopt. Zo snel dat TIME als vervanger van een wachtlus zoals: FOR DELAY=1 TO 500:NEXT DELAY, toch niet zo geschikt is, als men soms wil aanbevelen. Wie een 64 Kb machine heeft, beschikt feitelijk over ruim 80.000 al dan niet zelf schakelbare bytes. Die zijn ook hard nodig. Een enkele letter gebruikt 8 bytes om met goed fatsoen op het scherm te kunnen verschijnen. Die verdeling in 8 bits van een byte zet zich door het gehele systeem voort. De acht is favoriet. We komen het nog wel eens tegen. U behoeft natuurlijk die kodes niet van het scherm af te noteren. Al is de handel ook in APENDIX-II ondergebracht, voor de hex-kode zal het snel geen noodzaak meer zijn hiervoor de lijsten te raadplegen.
Het eerste programma heeft z'n dienst gedaan. Loopt u de listing eens door. Wellicht zit er iets in, dat u kunt gebruiken. Er is gebruik gemaakt van screen2, omdat het grafische in dit geval de nadruk had. Aan dat wissen en herprinten van de teksten zal de konklusie verbonden worden, dat tekstgebruik in screen2 verre van ideaal is. Een wis-

funktie als SPACE$ kan niet gebruikt worden. Hiervoor moet gebruik worden gemaakt van LINE (X,Y)–(X,Y), schermkleur, BF. Eerst bekijken hoe gewist moet worden en nogal omslachtig ook. Hier ging dat niet anders. Er is een dankbaar gebruik gemaakt van de grafische instrukties van dit scherm. Loopt u niet al te gehaast door boek en programma's. Het is er ook, om u op iedeetjes te brengen voor eigen programma's. Hier mag dat. Meer algemeen getuigt het van de goede mentaliteit, om tussen mijn en dijn te blijven onderscheiden. Er is niets tegen het kopiëren van een programma, dat u hebt gekocht, mits het de opzet is van zo'n kopie de werkcassette te maken om het origineel veilig weg te kunnen bergen. Dat is zelfs aan te bevelen. Niet alle cassette's zijn beveiligd tegen het per ongeluk wissen. Behalve de PLAY-toets voor het CLOADen is in een onbewaakt moment snel ook de REC-toets ingedrukt en daarmee kan een wellicht duur betaald programma nog duurder worden. Echter, zo'n kopie al dan niet tegen betaling aan derden ter beschikking te stellen, dat mag u in elk geval niet overkomen. Dat is meer dan alleen pronken met andermans veren. Het is gewoon ordinaire diefstal. Van software dat veelal ten koste van veel tijd en geld tot stand is gekomen. Zulk ongeoorloofd kopiëren gebeurt veel te vaak. Daarbij realiseert men zich niet, dat producenten er op de duur letterlijk geen brood meer in zien om software voor de MSX op de markt te brengen. Zo royaal is het aanbod van meer funktionele programma's toch al niet. Inplaats van een "kick" te krijgen bij een geslaagd kraken van andermans programma's, zult u dit beleven aan de eigen produkten. Dat is echt eervol. Maar goed, laten we het hebben over een andersoortig karakter. Dat van de letters, cijfers, leestekens en symbolen. Die hele, uitgebreide MSX-karaktersets.

Met opzet is "sets" geschreven. Want die voorkeur voor de 8 speelt ook hier een rol. De MSX-computer heeft feitelijk 32 sets á 8 karakters in huis. Dit komt niet tot uitdrukking in bijvoorbeeld Basic-instrukties, omdat daarin niet met setnummers wordt gewerkt. Toch is aan te bevelen, om de karakters te onderscheiden naar hun sets. Wie er veelvuldig mee omgaat, zal tenslotte aan een set de karakters weten te herkennen. Overbodig is dat zeker niet. Kleurverandering kan uitsluitend per set van 8 karakters plaatsvinden. In zo'n VPOKE-formulering is dan met het eerste karakter van een set te beginnen en met de laatste te eindigen. Als het om die ene set gaat althans.
In APPENDIX-I zijn alle karakters met hun kodenummer in hun sets geplaatst. Dat maakt het gemakkelijker, om een eerste en eventueel laatste kodenummer in een set vast te stellen. Bij deze karakters gaat het overigens slechts deels om de ASCII-sets. Deze standaardsets beperken zich tot de setnummers 5 tot en met 16. Van spatie (kode 32) tot delete (kode 127) omvatten ze alle gebruikelijke letters, cijfers,

leestekens en nog wat symbolen. Daarmee is rekening te houden bij het gebruik van MERGE. Kodenummers lager dan 32 en hoger dan 127 worden niet als gewone ASCII-kodes beschouwd. Daarom zijn de eerste 4 en de 16 sets ná kode 127 extra's, die de MSX mede tot een plezierige machine maken. Dit ASCII staat overigens voor American Standard Code for Information Interchange. We kennen ASC ook als Basic-instruktie. Het geeft het kodenummer van het aangegeven karakter. Maar dat wist u al.

Aandacht voor deze karakterset is nodig in direkte relatie met het hier te bespreken onderwerp. Echter ook bij puur tekstgebruik zullen bijvoorbeeld accent-letters niet overbodig zijn. Ze zitten er allemaal in vanaf set 17. De kwestie is alleen: hoe krijgen we ze er uit. Als we het dan toch over de basis van Basic hebben, dan is dat daarvan wel een heel essentieel aspect. In handleidingen wordt in het gunstigste geval nog onthuld, hoe we de karakters van set 1 t/m 4 op het scherm kunnen krijgen. Dat is, behoedzaam uitgedrukt, nogal merkwaardig. We moeten zelf maar uitzoeken welke toetsen in te drukken zijn om een karakter vanaf kode 128 op het scherm te krijgen. Bijster vlot werkt dat niet natuurlijk. Daarom is bij de sets in APPENDIX-1 ook aangegeven welke toetsen te drukken zijn om een karakter op het scherm te produceren. Of dit in alle gevallen ook voor uw computer geldt, dat zult u proefondervindelijk moeten ervaren. Vermoedelijk wel gezien de standaardisering. Hoewel . . . kode 127 bijvoorbeeld behoort leeg te zijn; het wordt voor delete gebruikt. Bij mijn machine is dat niet het geval en is kode 201 deze betekenis gegeven. Het handigste is nog, om met: ?A$=CHR$ (kodenummer) het betreffende karakter te laten weergeven en dan via de aangegeven toetsen dit karakter op het scherm te zetten. Klopt het, dan zitten we goed. In het andere geval zult u andere toetsen moeten proberen. Totdat het bewuste karakter gevonden is. U kunt dan de voor uw apparaat geldende toetsen-kombinatie in APPENDIX-1 noteren. De meeste karakters zijn op het scherm snel te herkennen. Bij anderen, zoals de kodes 192 t/m 223 is dat lastiger. Hoe ze eruit dienen te zien, dat valt eveneens in APPENDIX-1 te konstateren. Met de cursor en de letters van de ?A$ etc. regel moet worden bepaald om welk karakter het hier precies gaat. Het is veel belangrijker dan u wellicht meent.
Om zonder sprites toch figuren op het scherm te plaatsen, dienen deze gePRINT te worden. Dan moeten we gewoon weten, hoe de karakters achter een PRINT in te tikken zijn. Daarmee zult u veel te maken krijgen bij de toepassing van VPOKE. Zelf heb ik deze lijsten met karakters, kodenummers, setnummers en in te drukken toetsen in gekleurde hoesjes gedaan met een kartonnetje ertussen. Zo zijn ze steeds bij de hand en is het gezochte snel te vinden. Wellicht een idee voor u. Dui-

delijk zal het in elk geval spoedig worden, waarom wij zo'n hulpmiddel binnen handbereik dienen te hebben.

In beginsel kan elk karakter van die 32 sets veranderd worden in een door u zelf te ontwerpen figuurtje. Het is alleen niet zo wijs, om daarvoor een karakter uit de ASCII-sets te gebruiken. Die A te veranderen in bijvoorbeeld een zwart blokje, dat werkt niet al te prettig bij het nalopen van ons programma. Want wat we eenmaal veranderen, dat blijft in die gestalte totdat de computer uit/aan wordt gezet. Alleen wanneer alle extra sets zijn gebruikt, dan kan nog een ASCII-set worden aangesproken. Bijvoorbeeld set 16 met toch zelden gebruikte karakters. Vanzelfsprekend gebruikt u hiervoor nooit set 5. Noch voor kleur, noch voor een verandering van een karakter. Die set begint immers met kode 32 en dat is onze spatie. Daarmee is het scherm altijd overladen, zolang er niet iets anders staat. Geef set 5 bijvoorbeeld de kleur rood en het gehele scherm wordt rood, ongeacht wat u in COLOR hebt aangegeven. Alleen wie snel bijvoorbeeld het scherm voorzien wil van lijntjes, kan zo'n lijntje in de adressen van kode 32 VPOKEn. Er zal alleen amper behoefte voor bestaan. Want al die lijntjes komen óók terug op alle plaatsen in het programma waar eerst een spatie stond. Daarvan dient men een liefhebber te zijn.

Behalve bij tekstverwerking lenen sommige karakters uit die extra sets zich heel goed voor direkt gebruik. Met de lijntjes bijvoorbeeld als de kodes 16 t/m 31 zijn heel goed kaders te fabriceren in screen1. Ze kunnen LINE tot op zekere hoogte vervangen. Veel van de andere karakters hebben eigenlijk geen direkt praktisch nut. Ze zijn er voornamelijk om de adressen te vullen. En daardoor om door ons veranderd te worden in eigen karakters. En daarmee belanden we bij het feit waarom het hier gaat. Het gebruik van vooral VPOKE in onze programma's. Over POKE en PEEK kunnen we kort zijn. Te POKEn zijn uitsluitend machine-bytes van machine-kode en hier en daar om hooks in het systeemgebied om te buigen. Verder valt er met POKE niets te doen. Deze instruktie kan maar het beste met rust gelaten worden. PEEK richt geen schade aan. PRINT PEEK(adres) brengt alleen op het scherm, wat er in dat adres is opgeslagen.
Dat komen we weldra tegen. Hetzelfde is weliswaar met VPEEK het geval, doch daarmee is nog zo het één en ander meer te doen. Het goede gebruik van VPOKE en VPEEK behoort geheel tot de basis van Basic. Zonder deze instrukties blijft het allemaal erg beperkt. Wie er evenwel niet mee vertrouwd is, weet er evenmin iets zinnigs mee aan te vangen. In direkt verband hiermee kennen we nog BASE (X) en daarvoor geldt hetzelfde. Wat kunnen we ermee? Heel veel! Daarmee brengen we kleur en dekoratie op het scherm. En als we het over stief-

kinderen hebben, dan staat VPOKE helemaal bovenin de lijst. Wanneer er in handleidingen en vaak ook in MSX-Basic-voor-beginners boeken over gesproken wordt, dan blijft nadere uitleg veelal achterwege. Met de dooddoener dat het gebruik ervan zonder grondige kennis van de opbouw van het geheugen uit de boze is. Het wekt soms de indruk dat de publicist in kwestie zelf niet weet, wat hij ermee moet. Want al is enig inzicht aangaande die opbouw zeker nodig, met dat grondige valt het nogal mee. We hoeven er niet eerst een speciale opleiding voor te volgen. Bovendien gaat het hier om het adressenblok van VRAM. Al VPOKEn we mis, de machine zal er zeker niet door blokkeren. Zoals dit met POKEn in RAM wel snel het geval kan zijn. Vanzelfsprekend is het wel zo prettig om te weten, waarmee we bezig zijn als er ge-VPOKEd moet worden. Alleen dan verkrijgen we het gewenste resultaat. Laten we daarom vervolgen met een rondreisje door dat geheugen. Een tripje langs de elektronische monumentjes van de MSX.

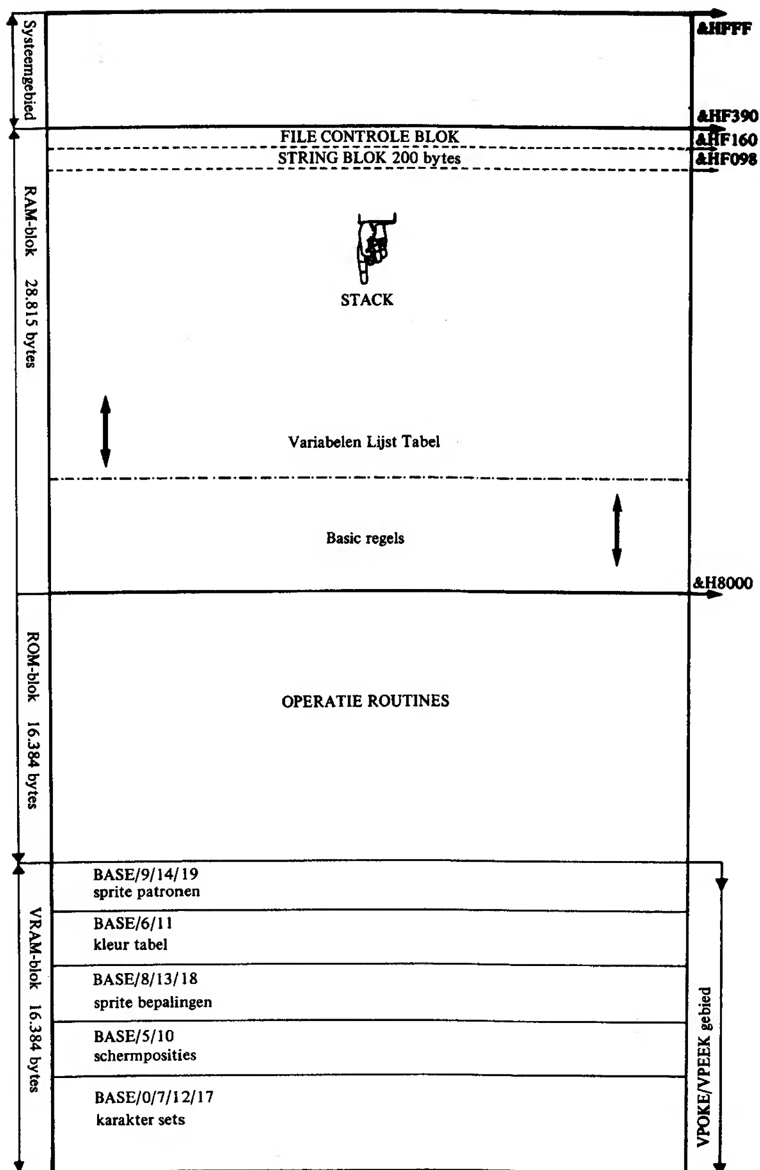
Het tweede programma komt in RAM, als u de listing van het programma PEEK gaat intypen.
Dat programma wijkt wat af van soortgenoten. Vooral onze tocht door ROM zou tamelijk saai worden, wanneer er niets dan byte-waarden op het scherm zouden komen. Daarom zijn 7 van de 67 operators, waarvan in ROM gebruik wordt gemaakt, in dit programma opgenomen. Zo'n operator is een machine-instruktie zoals PRINT dat is in Basic. Veelal hoort daar een OPERAND bij. Deze kan zowel een voorwaarde als een adres of beide bevatten. Voor machine-kode programmeurs bestaat er een speciaal taaltje, om deze operators en operands boven hun bytewaarden uit te tillen: mnemonic. Anders zou het werken er mee een dagelijkse hersenspoeling in hexadecimaal badwater nodig maken. Al zijn die 7 mnemonic-kodes dan gebruikt in het programma, u kunt er uiteraard verder niets mee doen. Het is er uitsluitend ter oriëntatie. Voor een tikkeltje meer notie inzake de routines in ROM. Ze duiken soms ook op, waar ze helemaal niet behoren te zijn. We dienen dat voor lief te nemen.

Zo kunt u JP op het scherm zien verschijnen. Dat staat voor JUMP en is vergelijkbaar met GOTO. Wat direkt na zo'n JP volgt aan bytes kan een adres zijn.

Voorstelling geheugenopbouw

Ca 3000 bytes van het RAM gebruikersgebied zijn gereserveerd voor het systeemgebied, dat bij de uitvoering van de opdrachten wordt gebruikt. De overige 28.815 bytes zijn evenmin geheel beschikbaar voor de opslag van Basic-regels.
Bovenin is ruimte gereserveerd voor het File Controle Blok. Normaal bestemd voor 1 file. Het formaat hangt af van het gebruik van de instrúktie MAXFILES. Direkt hieronder bevindt zich het stringblok, dat via CLEAR kan uitbreiden. Echter ook gereduceerd kan worden door bijvoorbeeld CLEAR 50 als weinig of geen stringvariabelen in een programma worden gebruikt. Tenslotte heeft de Stack ruimte nodig. Zonder wijzigingen via MAXFILES en/of CLEAR begint de Stack op adres &HF098. De Stack groeit omlaag in het geheugen. Bij een lang Basic programma kan aldus de foutmelding STACK OVERFLOW op het scherm komen.
Vanaf adres &H8000 worden de Basic-regels volgens machinekodes in opeenvolgende adressen geplaatst. Direkt boven de zo opgeslagen laatste Basic-regel bevindt zich de VLT. Een lijst van in het programma gebruikte variabelen. Deze groeit met het Basic-programma mee. Een regel meer doet de VLT omhoog toenemen een regel minder omlaag afnemen. Behalve dat aldus de nodige bytes afgeknabbeld worden van de beschikbare 28.815 bytes is er gedurende de uitvoering van een programma veelal geheugenruimte nodig om bijvoorbeeld teller-waarden bij te houden. Daar is bij lange programma's rekening mee te houden. Het is eenvoudig te kontroleren door vóór en na het RUNnen ?FRE (0) te typen.
In het Rom-blok zijn alle routines voorgeprogrammeerd. VPOKE noch POKE kan daarin enige wijziging brengen. VPOKE/VPEEK is alleen in het VRAM-blok te gebruiken. ROM (Read Only Memory) geeft het één en ander al aan. Zo ook RAM (Read Acces Memory) als het tijdelijke opslaggebied voor Basic-regels en machinekode. Bij het uitschakelen van de computer worden deze adressen ontdaan van die tijdelijke opslag. Bij het aanzetten worden de adressen gevuld met &H00 en &HFF bytes en zijn zij weer gereed om nieuwe Basic-regels op te slaan. Het gebruik van NEW wist alleen de Basic-instrukties uit het geheugen. De eventueel gebruikte machinekode wordt hierdoor niet aangetast. Voor Basic is alleen het VRAM-blok van belang.

Wij zouden in Basic een regelnummer gebruiken. Dat adres is dan alleen in omgekeerde volgorde weergegeven. Gesproken wordt van LOW-BYTE en HIGH-BYTE. De tweede is te lezen vóór de eerste. Zou aldus na JP bijvoorbeeld als eerste byte DF volgen en daarna als tweede 07, dan kán het gaan om adres &H07DF. Het is alleen dat u het


Systeemgebied
RAM-blok 28.815 bytes
ROM-blok 16.384 bytes
VRAM-blok 16.384 bytes
&HFFF
&HF390
&HF160
&HF098
FILE CONTROLE BLOK
STRING BLOK 200 bytes
STACK
Variabelen Lijst Tabel
Basic regels
&H8000
OPERATIE ROUTINES
BASE/9/14/19
sprite patronen
BASE/6/11
kleur tabel
BASE/8/13/18
sprite bepalingen
BASE/5/10
schermposities
BASE/0/7/12/17
karakter sets
VPOKE/VPEEK gebied

maar even weet. Van enig belang mag het hier verder niet zijn. Hetzelfde geldt voor mnemonic CALL. Dat is vergelijkbaar met GOSUB. Wel kennen we ook een Basic-instruktie CALL, maar die is voor iets anders gemeend. Als u CALL op het scherm ziet komen, dan weet u, dat daarmee een andere routine in ROM wordt opgewekt het zijne bij te dragen aan de uitvoering van een programmadeel. Dan is nog NOP (Not OPeratable) gebruikt en die zult u dikwijls tegenkomen. Veelal vormt het een afscheiding tussen twee onderdelen van een routine. Samen met RST, ook opgenomen, wordt deze 0-byte na het aanzetten van de computer in groepen van 8x8 in RAM gezet. In het niet voor het Basic-programma gebruikte deel van RAM zult u dan ook schier eindeloze reeksen &H0 en &HFF bytes tegenkomen ofwel NOP en RST. Als operator is deze RST overigens een speciale vorm van CALL. Aangeduid als "zero-page-adressing." Kunt u gelijk weer vergeten. Verder zult u nog RET kunnen tegenkomen. Precies. Dat staat voor RETurn. Zo'n RET sluit een routine af. Tenslotte zijn de mnemonics OUT en IN gebruikt, die wijzen op een schrijven of lezen van de videopoort. Daar moeten de burgers van ROM aldus dat beroep doen op het fijne gezelschap in VRAM of om geluid te laten produceren. Op deze wijze zal ROM iets meer tot de verbeelding kunnen spreken. Wilt u er alles van zien in mnemonic, dan is de aanschaf van een assembler noodzakelijk. Onmisbaar bij het programmeren in machinekode. Helaas wél met een soms schier onbegrijpelijke handleiding.

Bij ons reisje door ROM zult u nu en dan ook op zichzelf staande letters en dergelijke zien opduiken. Dat is verder zonder betekeinis. CHR$ is alleen opgenomen, omdat er in ROM ook teksten zijn ondergebracht. En wat daar dan staat, dat wordt zo handiger leesbaar voor u. Zover zijn we echter nog niet. Voor ons thema is het VRAM-blok veel belangrijker. Laten we daarmee dus beginnen. Het start eveneens met adres 0, omdat het een opzichzelf staand adressenbestand is. In het programma is geen gebruik gemaakt van INKEY$. Voor de regels 50 en 60 is simpelweg REM gezet. Daar start het PEEKen van ROM en RAM. Als we zover zijn, dan haalt u REM gewoon weg en plaatst u REM voor de regels 30 en 40. Een fluitje van een cent. Na RUN verschijnen de eerste 8 adressen van het VRAM-blok. Nog niet om over naar huis te schrijven, zult u wellicht denken. Het gaat echter wél om kodenummer 0 van karakterset 1. We staan aan het begin van de 2048 adressen, die starten met BASE(7), (12) en (17).
De opslagplaatsen van de MSX-karaktersets. Die zijn overigens uitgebreider dan die 2048 adressen. Tekstscherm screen0 beschikt over een eigen serie sets met eveneens 2048 adressen, startend via BASE (2) ofwel adres 2048. Daarop volgen nog eens 2048 ongebruikte adressen. Daarin kan weliswaar nog een komplete serie van 32 sets worden

ondergebracht, maar het is niet mogelijk deze op het scherm te krijgen. Aangenomen wordt, dat dit derde adressenbestand gereserveerd is voor de opslag van afwijkende karakters van een tweede taal naast het engels. Bijvoorbeeld japanse of arabische schrifttekens. Om die op het scherm te zetten, is dan wel een speciale Basic-instruktie nodig.

De INPUT-regel maakt het u mogelijk om óf de volgende serie adressen op het scherm te zetten, óf om een ander adres in te tikken. Het hoogste adres in VRAM is evenwel 16384. Wilt u geen foutmelding riskeren, dan dient u daar onder te blijven. Drukt u alleen RETURN, dan verschijnen de adressen achtereenvolgens op het scherm mét het daarin opgeslagen karakter. Wat er hier onder BYTE en DEC. staat, is de opbouw van zo'n karakter. En zoals u ziet, zijn daarvoor steeds 8 adressen of bytes nodig. Indien u een 8x8 matrix bij de hand hebt, dan kunt u een karakter volgens de weergegeven hex-kode natekenen. Een onzinnige bezigheid hoeft dat niet te zijn. Daarmee oefent u zich in het gebruik van de hex-kode én u kunt er achter komen hoe bijvoorbeeld een ronding ingetekend kan worden. Maar het is eveneens ondergebracht in APPENDIX-I. Neust u de zaak eens door tot en met kode 255 van de cursor. Die kunt u overigens wél van kleur maar niet van vorm veranderen. De uitzondering onder al deze karakters. De BASE (x) instruktie is er alleen voor het startadres van een bepaald adressenbestand. In plaats van: A=BASE (7) zouden we aldus ook: A=0 kunnen gebruiken. Het gebruik van BASE hoort niet alleen bij VPOKE/VPEEK, het is doorgaans ook eenvoudiger. Zo, beter dan A=14336 zal A=BASE (9) zijn. Terwijl we in de praktijk meestal ook een bepaalde waarde bij deze A tellen om vanuit een hoger adres te kunnen starten. Dán vooral is het gebruik van: A=BASE (x) + X wel zo handig. Het komt nog.

In de adressen 2048 t/m 4095 zitten aldus de karakters voor screen0. Daarvan wordt ook gebruik gemaakt in de file-teksten voor screen2 en 3. Wagen we ons nog dieper in VRAM, dan komen we tussen adres 4096 en 6143 niets dan &H0 en &HFF tegen. Het zijn die ongebruikte adressen waarover we het al even hadden. Interessanter wordt het weer vanaf adres 6144 t/m 6911. Hier storten we ons in de opslagplaatsen van het scherm. Daarvan raakt u snel genoeg overtuigd, als u even blijft volharden in het RETURNen. Wat die decimale kode 32 te maken heeft in die eerste adressen? Dat behoort u al te weten. Het is de spatiekode.
Daar staat verder niets op het scherm. Misschien wat lastig dit kijken op het scherm en het lezen van deze woorden. Ik blijf echter op deze goede samenwerking vertrouwen. Wat kan ik anders doen? Bent u eenmaal bij adres 6240 gekomen, dan zal het opeens duidelijker kunnen

worden. Hier wordt weergegeven, wat er op het scherm staat. En Adres: is het eerste dat er staat. Dit adressenbestand wordt aangegeven als "name-table". Maar hier mag "schermregistratie" het beter doen. BASE(5) en (10) geven het startadres. Wat er ook op het scherm aanwezig is, met uitzondering van de sprites, het wordt allemaal in deze adressen geregistreerd volgens het bij een karakter horende kodenummer. Dat is alleen waardevol voor ons bij het gebruik van screen1. Middels BASE (10) voor screen2 worden er uitsluitend schermposities vermeld. Niets van wat er zich daarop bevindt. Mocht het vermoeden bij u zijn opgekomen, dat van zo'n schermlezing wel eens een dankbaar gebruik zou kunnen worden gemaakt, dan bent u bezig onder mijn duiven te schieten. Het moet nog komen. Laten we voortgaan met het tripje door VRAM.

Het volgende adressenbestand dat we tegenkomen kent BASE (8), (13) en (18) als startadres. Het omvat de adressen 6912 t/m 7039. Hoewel aangegeven als "sprite-attribute-table" dekt hier "sprite-bepalingen" de lading beter. We zien steeds groepen van 4 adressen met identieke waarden. Dat zal geen verbazing wekken, want er zijn geen sprites gedefinieerd, die via PUT SPRITE bepalingen hebben ontvangen. Er zijn dus 4 bepalingen mogelijk. De eerste is de Y-positie van een sprite. Die is hier de waarde 209 gegeven. Het gebruik hiervan resulteert in het verdwijnen van een sprite. Kennelijk hebben de ontwerpers van MSX het zekere voor het onzekere willen nemen. De tweede betreft de X-positie van de sprite. Hier aldus met een 0 gehonoreerd. De derde bepaling is die van het vlaknummer. Niet van het spritenummer zoals weleens wordt beweerd. Het loopt van 0 t/m 31. Dit vlaknummer is bepalend voor de volgorde waarin sprites op het scherm komen en van veel belang bij animaties. Dat spritenummer is van minder belang. Het wordt bepaald door de volgorde, waarin gedefinieerde sprites in de adressen van de sprite-patronen terecht komen. Een plane of vlaknummer kan nooit hoger zijn dan 31. Er worden bij het gebruik van sprites niet meer dan 32 transparante lagen of stroken over het scherm gelegd. Wat we nu op het scherm zien, dat betreft evenwel de 8x8 sprites.
Zou er met het 16x16 formaat worden gewerkt, en het is of het één of het ander, dan begint het vlaknummer eveneens met 0, doch loopt steeds met 4 op tot en met vlaknummer 124. Dat is echter alleen, om de adressen in deze modus te zetten. De vlaknummering bij ook grote sprites blijft lopen van 0 t/m 31.
Een sprite dubbel zo groot maken middels screen1,1 of screen1,3 heeft verder geen invloed op deze adressen. Dat is een klus voor de VDP. Daarom kunnen we inplaats van screen 1,X ook gebruiken:

```
VDP (1) = 224    ; voor 8x8 sprites
VDP (1) = 225    ; zelfde maar dubbel zo groot
VDP (1) = 226    ; voor 16x16 sprites
VDP (1) = 227    ; zelfde maar dubbel zo groot
```

Maar waarom zouden we dit doen, indien daarvoor geen grondige reden bestaat? De formaatbepaling bij screen is gewoon korter. De vierde en laatste bepaling betreft de kleur. Hier kode 15 voor wit. Het zijn gewoon standaardwaarden. Pas als we met sprites werken, komen de aangegeven waarden in deze adressen te staan. Bijvoorbeeld:

```
PUT SPRITE 0, (10,25),1 , 0
```

Dan gaan die eerste vier adressen hier er zo uitzien:

6912	25	(de Y-waarde)
6913	10	(de X-waarde)
6914	0	(het vlaknummer)
6915	1	(kleurkode voor zwart)

Strikt genomen houdt dit in, dat we de PUT SPRITE instruktie eigenlijk niet nodig hebben. We zouden de waarden óók direkt in deze adressen kunnen VPOKEn. Eveneens als we zo'n sprite over het scherm willen laten bewegen. Daarvoor VPOKEn we gewoon oplopende of afnemende waarden in adres 6912 of 6913. Het gebruik van PUT SPRITE is echter handiger. Daarmee is ook het spritenummer aan te geven en eventueel kan gebruik worden gemaakt van de overigens niet al te zinnige STEP-funktie.

Nadat u hiervan breedvoerig kennis hebt genomen, zetten wij ons reisje voort door nog steeds VRAM. De adressen tussen 7040 en 8191 zijn ongebruikt. Er zou daarin nog ruimte zijn geweest voor 72 8x8 sprites. Wellicht is het daarvoor achter de hand gehouden. Ik weet het niet. De juiste bedoeling van de ontwerpers wordt duidelijker vanaf adres 8192. Dat kent BASE (6) en (11) als startadres en betreft de opslag van de kleurwaarden. Ofschoon deze adressen oplopen tot 14335 hebben we vooral met de eerste 32 te maken tot en met adres 8223. De overige zijn ongebruikt. Vanaf adres 8192 wordt aldus 32 keer de kleurwaarde &H1C of decimaal 28 weergegeven. Dat is de in COLOR bepaalde kleur; zwart op donker-groen. Deze COLOR bepaling wordt trouwens ook in dat systeemgebied bijgehouden. Dat het hier om slechts 32 adressen gaat, behoort u niet meer te verwonderen. Immers, er zijn even veel karaktersets in onze MSX. Zo kent elk van deze sets aldus een eigen kleurbepaling.

Hier blijkt dat uiteraard niet, omdat COLOR de hele zaak in gelijke kleuren penseelt. Pas als we een bepaalde set via VPOKE van kleur hebben gewijzigd, zal die nieuwe kleurwaarde in het betreffende adres te lezen zijn. Het mag vooralsnog duidelijk maken, dat we desgewenst alle 32 sets een andere kleur kunnen geven. Wél zijn er slechts 16 kleuren beschikbaar, maar daarmee tegelijk 256 kleurenkombinaties. Daarmee zou u alvast kennis kunnen maken in APPENDIX-III. We kunnen er echter beter nader op ingaan wanneer we het verderop over screen3 zullen hebben. Het typische kleurenscherm. Al die adressen na de eerste 32 zouden best eens gebruikt kunnen worden voor dit scherm. Het is alleen lastig te kontroleren, omdat in screen3 niet al te veel tekst kan worden weergegeven. Maar het aantal van de 6114 onbenutte adressen hier komt vrij aardig overeen met de schermlokaties van screen3. De kleuren in screen2 staan hiervan los. Als we PAINT of LINE/BF gebruiken dan wordt het betreffende deel van de schermkleur veranderd. Zou dit anders zijn, dan was er geen tekst op zo'n kleurvlak te PRINTen zonder dat de achtergrondkleur zou worden gewist. Immers, voor screen2 behoeven we de achtergrondkleur voor tekst niet aan te passen. Deze kleuradressen, althans de eerste 32, zijn medebepalend voor het aanzien van ons scherm. We zijn geenszins aan COLOR gebonden.

De trip door VRAM nadert het slot. We komen bij adres 14336. Dat nummer kunt u gauw even intikken. Het is de startwaarde verbonden met BASE (9) - (14) en (19). Niets dan lege adressen. Bedoeld om in groepjes van 8 de patroonwaarden onder te brengen, die wij zullen hanteren bij het ontwerpen van een sprite-figuur. We zitten aldus in de spritepatroonadressen. In plaats van over 8x8 sprites kunnen we algemeen beter spreken van een 8x8 of 16x16 matrix. Voor te stellen als een vierkant met 8 lagen met elk weer 8 hokjes. Of 16 natuurlijk. Het is beter, omdat het bij het ontwerpen van figuurtjes helemaal niet om sprites hoeft te gaan. Veel vaker zullen we zo'n ontwerp gewoon met PRINT op het scherm zetten. Vandaar dus dat de benaming "sprite-editor" voor al dan niet professionele software argwaan wekte ten aanzien van de bekendheid met de basis van Basic.
Zo zou het niet zijn genoemd, indien er kennis had bestaan omtrent het PRINTen van figuratie op het scherm. Maar goed. Blijven we bij de les. Deze laatste VRAM-adressen bieden ruimte aan 256 te ontwerpen sprites formaat 8x8 of 64 in formaat 16x16. Aldus kunnen spritenummers gaan van 0 t/m 255 of van 0 t/m 63. Weliswaar kunnen er "slechts" 32 sprites tegelijk op het scherm, doch door de vlaknummers aan hogere spritenummers te schenken, zal tenslotte ook nummer 255 zijn entree kunnen maken. Het is bedoeld, om bij veel verschillende sprites deze alvast te definiëren om ze bij de hand te hebben in het geval dat.

Hoeveel we er ook willen vastleggen, deze mogelijkheid om een sprite-ontwerp eigen adressen te geven, is zondermeer een groot voordeel dat ons MSX hier biedt. Bij mijn oude computer en bij vele anderen moest daarvoor een karakterset worden aangesproken. Een grote beperking. Nu wordt geen enkel karakter aangetast terwille van sprites. Ze hebben hun eigen adressen, hoe tijdelijk de bewoning dan ook mag zijn. Werkelijk een groot voordeel.

Hoewel onze sprites naderhand nog hun speciale aandacht krijgen, kunnen we daarop in dit verband alvast even vooruitlopen. Misschien is het al duidelijk geworden, dat de SPRITE$-instruktie in Basic feitelijk een onding is. Het vergt een veel te omslachtige definiëring van de patroonwaarden. Veel eenvoudiger én direkter is het, om deze waarden in de sprite-patroon-adressen te VPOKEn. Alvast een voorbeeld:

```
A=BASE (9)
For I=0 TO 7
READ C$
VPOKE A+I, VAL ("&H" +C$)
NEXT I
```

Dat is alles. En al zou u 256 8x8 sprites in de adressen willen laden, dan hoeft u na FOR alleen 0 TO 2047 in te tikken; 8 adressen voor elke sprite. Vanzelfsprekend dienen er dan ook de DATA-regels te zijn met deze 256x8 byte-waarden. Inplaats van hex is decimaal te gebruiken. Dan wordt het READ C en is VAL en "&H" + niet nodig. Zou het echt om 256 sprites gaan, dan spaart u met dat hexadecimale wél heel wat bytes uit! Er is geen zinnige reden te bedenken om deze hex-kode niet te bezigen. Daarom is het wat merkwaardig dat toch kundige programmeurs evengoed de decimale hanteren. U kunt dat regelmatig tegenkomen in de listings, zoals gepubliceerd in de MSX-magazines. Die werkwijze houdt simpelweg in, dat zij voor elke patroonwaarde een lijst moeten raadplegen om er de decimale kode van te kunnen vaststellen. Wie de hex-kode eenmaal kent, weet het gewoon uit z'n blote hoofd. U gaat ook tot deze behoren.

hoofdstuk 3
Boemelen door ROM en RAM

Video Read Access Memory ofwel VRAM hebben we gehad. Dat "access" wil duidelijk maken, dat gelezen wordt, wat we er al dan niet zelf in onderbrengen. Vandaar ook: Read Only Memory ofwel Rom. Alleen wat door de ontwerpers in al die schakelaartjes is vastgelegd, kan worden gelezen en wij kunnen daar niets aan veranderen. Gelukkig niet. Als we naar willekeur waarden zouden kunnen POKEn in ROM, dan zou geen Basic-instruktie meer fatsoenlijk kunnen worden uitgevoerd.
Wellicht zou dan de mededeling op het scherm komen: ZOEK HET ZELF MAAR VERDER UIT! We reizen alleen door ROM uit pure nieuwsgierigheid. Om tenminste een indruk op te doen hoe het er daar uitziet. Echt nodig hebben we het niet. Het gaat ook wat langer duren dan bij VRAM. Breekt u het Peek-programma aldus af met CTRL-STOP en zet Rem voor de regels 30 en 40. Dat Rem voor de regels 50 en 60 haalt u nu dus even weg. Anders werkt er niets meer. Als u zover bent, dan gaan we RUNnen. Helemaal rechts ziet u nu Mnem, voor mnemonic staan. En meteen al een JP. Vanuit dit adres kan aldus een GOTO worden uitgevoerd. Hier gaat het trouwens om de adressen bedoeld voor operator RST. Niet van veel belang voor ons. Laten we daarom eerst eens de adressen intikken, waarin voor ons direkt leesbare zaken zijn opgeslagen.

Geef als INPUT adres 5033 en RETURN door tot adres 5182. Inderdaad, hier houdt men in ROM de teksten van de funktietoetsen opgeslagen. Hoe is dat mogelijk? Immers, we kunnen in ROM niets veranderen en tóch via KEY X," " een andere tekst aan een funktietoets toekennen. Deze teksten worden na het aanzetten van de computer gekopieerd in het systeemgebied. Dat is te bewijzen, zodra u adres 63615 hebt ingetypt. Tot aan adres 63764 zitten die funktietoets teksten. Keren we evenwel terug in ROM. U hebt gelijk. Wat na het aanzetten van de computer op het scherm komt aan tekst inzake copyright en vrije bytes, dat is evenzeer in ROM opgeslagen. Tikt u adres 32472 maar in. Het gaat door tot adres 32550. Gezien? Dat weten we dan ook weer. Idem dito voor de foutmeldingen. Na INPUT van adres 15734 RETURNt u even door tot nummer 16356. Hier

haalt men vandaan, wat ons bij het programmeren soms zo kan ontmoedigen. Evengoed een goede zaak die foutmeldingen. Wie in machinekode programmeert moet het zonder doen. Dan loopt het of het loopt niet. Basic-gebruikers zijn hier bevoorrecht. Want al staat dit Basic dan voor: Beginners All-purpose Symbolic Instruction Code, het is zeker voor MSX inmiddels een volwaardige computertaal geworden. Al trekken bijvoorbeeld Pascal-gebruikers er hun neus voor op. U zult wellicht de foutmeldingen hebben gemist, die horen bij het gebruik van een diskdrive. Ze zitten er niet in. Vanuit die drive komen ze op het scherm. Weliswaar kunnen we aan de niet gebruikte foutmeldingnummers 72 t/m 255 eigen foutmeldingen toekennen bij gebruik van ERROR, doch zulke eigen verzinsels komen niet in ROM. Men heeft het er niet zo begrepen op vreemdelingen. Persoonlijk vind ik dit gebruik van ERROR wat overbodig. We hebben het niet nodig om de gebruiker van ons programma te laten weten, dat hij zich aan het vergissen is. Dan doen we het moeilijker dan nodig is. Niet alle MSX-instrukties zijn even zinvol. Neem nu bijvoorbeeld dat KEY LIST. Zonder KEY OFF kijk je toch al de hele tijd tegen die teksten aan. Wie zal er dan nog behoefte aan hebben ze alle tien nog eens op een rij te zien? Ik bedoel maar. Overigens zit de foutmelding bij een verkeerde INPUT apart in de adressen 19259 t/m 19276. Hieraan is de INPUT-routine van ROM te herkennen. In de cassette-routine komen we terecht door adres 28927 in te tikken. Kan ik meteen mooi iets aan u kwijt, dat u misschien toch al wist. Dat Skip: (negeren) is handig te gebruiken. Want al behoort het ons niet te overkomen, in het geval we toch niet meer zouden weten onder welke naam we een programma op band hebben gezet, dan kan gewoon CLOAD "ZOEKER" bijvoorbeeld worden getypt. Naar die naam wordt dan gezocht. Omdat deze dan niet wordt gevonden, wordt achter Skip: keurig de werkelijke naam van het programma op het scherm gezet. Op die manier kunnen we onze slordigheid weer ongedaan maken. Daarmee is dan door te gaan tot en met het laatste programma op de cassette. Handleidingen geven allerhande truuks om een programma toch in RAM te kunnen laden, als we daarvan de naam even zijn vergeten. Wanneer die naam echter niet in een REM-regel is genoteerd, dan weten we nog niets. Dan kan het alleen onder een nieuwe naam opnieuw geCSAVEd worden.

De foutmelding "Undefined line X" zit in de routine, waarvan bij RENUM gebruik wordt gemaakt. Niet geloven? Tik adres 21850 dan maar in. Toch wel handig zo'n INPUT-mogelijkheid, nietwaar? Anders zou je uren kwijt zijn om langs al die adressen te lopen. Het kan overigens gebeuren, dat tijdens de reis het scherm opeens wordt gewist en die INPUT-regel ook helemaal bovenaan komt te staan. Dan hebben

we met een CLS-routine of byte te maken. Die strikt zakelijk ingestelde figuren in ROM gaan er vanuit, dat we de boel gewist willen hebben. Dat willen we niet, maar er is niets tegen te ondernemen. Het herstelt zich vanzelf wel weer. Niets van aantrekken dus. De laatste min of meer leesbare mededelingen vinden we in de adressen 15411 t/m 15673. Min of meer leesbaar. Want behalve zakelijk zijn er ook krenterig in ROM. Wat we hier zien zijn alle Basic-instrukties inklusief die van de rekenkundige bewerkingen. Alleen zijn daarvan de eerste en laatste letter weggelaten. Om bytes te sparen. Kijken we even naar adres 15418. Wat te lezen is als EE behoort PEEK te zijn. In adres 15421 is het getal 23 ondergebracht. Het hoort bij PEEK, want het is het TOKEN-nummer van PEEK. Zoals 145 dat van PRINT is. Elke instruktie kent een eigen token-nummer. In Basic hebben we er niet al te veel mee te maken. Althans niet bewust. Echter, alles wat we aan Basic instrukties in programmaregels onderbrengen, dat wordt als even zovele token-nummers in RAM opgeslagen. Daar krijgen we nog even mee te maken. In deze adressen zijn deze tokens in alfabetische volgorde opgenomen gerekend naar de tweede letter van de instruktie. In APPENDIX-IV zijn ze allemaal in numerieke volgorde ondergebracht. Normaal hoort zo'n lijst niet thuis in een boek, waarin het om Basic gaat.
Omdat we echter toch bezig zijn wat inzicht te verwerven omtrent de diepten van de computer, willen we dat tenslotte afronden met het lezen van de Basic-regels, zoals dezen in RAM worden opgeslagen. Dan weten we overal een beetje vanaf. Het zou trouwens helemaal geen slechte gedachte zijn om een assembler aan te schaffen en u eens in machinekode te verdiepen. Niet om er meteen in te gaan programmeren, want dat is bepaald geen lichte taak, maar om de zaken in ROM nog wat beter te leren kennen zo tussendoor.

U ziet achter adres 15416, dat we nu op het scherm hebben (als het goed zit met onze samenwerking) de operator OUT vermeld. Die hoort hier niet. Ook andere operators kunnen opdagen, zonder dat ze zijn uitgenodigd. Net doen alsof u ze niet ziet. Keren we terug naar de lagere regionen van ROM. Dat kan als u adres 56 intikt. Daar staat een JP. En als u een tijdje op de RETURN-toets drukt, dan zijn er een 97 te tellen met elk twee adressen, waarin het spring-adres is ondergebracht. Nemen we de JP van adres 59. In goede volgorde gelezen staat er in goed Basic: GOTO &H049D, al gebruiken we in BASIC dan het decimale. Het is een beetje lastig. Adressen worden vaak in hex gegeven. BASIC kent geen instruktie om zo'n hex-kode te vertalen in een decimale. Andersom wél via HEX$. Wat we kunnen doen, is in een lus van 0 tot 65535 uit te vinden, om welke decimale waarde het bij een bepaald adres in hex gaat. Dat vergt alleen wel talrijke minuten. In een ander programma kunt u een routine aantreffen, waarin deze tijd is

teruggebracht tot maximaal 45 sekonden. Daar komen we het nog tegen. Waar we hier mee te maken hebben, zijn de CALLs, zoals de programmeur in machine-kode deze kan gebruiken. Je kunt zeggen, dat al die JP's de springplankjes zijn naar de routine in ROM. Omdat er vele routines in ROM zijn ondergebracht, is het veelal geen noodzaak deze nog eens ineen te knutselen in zo'n mnemonic-programma. We maken dan gemakshalve gebruik van wat toch al bestaat door middel van een CALL naar een bepaalde JP te gaan. Laten we maar even een voorbeeld pakken. Meteen een goede gelegenheid om te zien hoe een mnemonic-programma er uit kan zien. Hier een routine vergelijkbaar met VPEEK. Je bent aan het PEEKen of je bent het niet. Als u adres 74 intikt, dan is dat meteen ook het adres, dat de machinekodist aan zijn CALL kan geven. Het hoeft niet. Zo'n JP kan ook gepasseerd worden en meteen met CALL de routine worden opgeroepen. Na deze JP kunt u adres &H07D7 zien staan. In goed decimaals adres 2007.
Lopen we daar even heen middels onze INPUT. Wat u daar niet zondermeer kunt waarnemen, ziet er dan zo uit in mnemonic:

```
CALL &H7E       'een subroutine op adres 2028 moet helpen
EX (SP), HL     &H7EC: LD A,L   'L in A laden
EX (SP), HL     DI              'Disable Interrupt (tijd
                                 rekken)
IN A, (98H)     OUT (99H), A    'wat in A kwam naar de
                                 videopoort schrijven
RET             LD A,H          'nu high-byte in A stoppen
                AND 3FH         'de zaak vergelijken
                OUT (99H), A    'wegschrijven voor het
                                 scherm
                EI              'Enable Interrupt
                RET             'terug naar adres ná CALL
```

Het gaat hier niet om variabelen. Met die A wordt de accumulator aangeduid, het belangrijkste register van de micro-processor. En SP staat voor Stack-Pointer. We hebben het hier evenwel verder niet over machinekode. Het zal niet meer dan illustratie zijn van wat er zoal in ROM is ondergebracht. Dit voorbeeld behoort tot de kortste routines. Wanneer u door ROM wandelt, dan zal RET u er van tijd tot tijd op wijzen waar zo'n routine eindigt en waar een nieuwe begint. Zo is ook nog na te tellen hoeveel van die routines en subroutines er zijn. Veel langer willen we niet stilstaan bij ROM. Alvorens adres 32768 binnen te stappen als het eerste in RAM, toch nog even iets over die CALLs van de machinekodisten. In Basic is er evenzeer gebruik van te maken. Daarvoor hebben we DEFUSR en USR. Als u het PEEK-programma nu even afbreekt en zonder regelnummer intypt;

DEFUSR=&HCC:A=USR (0)

dan begint u te snappen, waarom er geen KEY OFF in het programma is opgenomen. Want als u RETURN geeft, dan verdwijnen de funktietoets-teksten als bij toverslag. En met: DEFUSR=&HCF:A=USR (0) komen ze terug. Op gelijke wijze is CLS te vervangen door: DEFUSR= &H6C: A=USR (0). Dit mag dan heel professioneel aandoen, maar CLS of KEY OFF/ON is gewoon korter en zinvoller. Anders wordt dit pas, indien er in Basic geen instruktie bestaat voor wat we willen realiseren. Zo zijn sprites bijvoorbeeld niet radikaal van het scherm te verjagen. Behalve door aan Y de waarde 209 toe te kennen. Bij 32 sprites op het scherm is dat nog een hele klus. Dán is het zinvoller om DEFUSR=&H69:A= USR (0) te gebruiken. De konsekwentie hiervan is wél dat naderhand die zo opgeblazen sprites opnieuw geVPOKEd moeten worden. Er zijn meer van deze "CALLs". Strikt genomen evenveel als er aan JP'S in die adressen te tellen zijn. We moeten er wél heel voorzichtig mee zijn.
De machine blijft weliswaar heel, maar een blokkering ontstaat snel en daar hebben we niet al te veel aan. Sommigen doen weinig anders dan dit experimenteren met zulke adres-definities. Alles is er nog niet voldoende van bekend, omdat de MSX een relatief nieuw ontwerp is. Het goede gebruik hiervan stelt wél een gedegen kennis omtrent de zaken in ROM als voorwaarde. Wij hebben dat hier niet nodig.

Eigenlijk zou nu NEW moeten worden gegeven. Dan zou het programma echter toch weer geladen moeten worden.
Slaan we daarom maar even een stuk RAM over en tikken we adres 34222 in. Het begin van de nog ongebruikte adressen. Bedoeld als opslagplaats voor Basic-regels. U kunt er even mee vooruit in het geval u per sé wilt weten, wat er allemaal in deze adressen zit. Niets dan &H0 en &HFF bytes en weergeven als NOP en RST. Dat wordt al te saai. Het PEEK-progrramma is onderdak geboden tot en met adres 34105. De regels zijn door NOPs van elkaar gescheiden. We komen er op terug. Na de laatste opslag volgen drie NOP-adressen. Daarna evenwel komt u de variabelen tegen, die in het PEEK-programma worden gebruikt. Wat er zo direkt boven de Basic-opslag adressen zit, dat wordt aangeduid met Variabelen Lijst Tabel ofwel VLT. Dat groeit met ons Basic-programma mee of zakt omlaag als we daaruit regels schrappen. Dit aldus variabele adres wordt bijgehouden in het systeemgebied. Zou u nu willen weten in welk adres het allerlaatste Basic-woord is opgeslagen, dat typt u:

?PEEK (63170) + 256*PEEK (63171)

Het aktuele adres van de VLT wordt gegeven en aldus drie adressen lager eindigt de Basic-opslag. Kan soms handig zijn om te kontroleren. Als ons programma flink gaat uitlopen en we bovendien nog machinekode in de hogere RAM-adressen hebben opgeslagen. Er zijn trouwens meer van deze al dan niet bijgehouden adresblokjes in ons RAM-gebied. Helemaal bovenaan direkt onder het systeemgebied zit het File Controle Blok. Gewoonlijk tussen de adressen &HF160 en &HF390. Het gebruik van MAXFILES doet de omvang ervan echter toenemen. Daar weer onder het String Blok met standaard 200 bytes. Het kan eveneens uitgroeien, als gebruik gemaakt wordt van bijvoorbeeld CLEAR 500. In het geval veel stringvariabelen worden benut in een programma. Aan dit CLEAR 500 kan een adres worden toegevoegd. Bijvoorbeeld: CLEAR 500,&HF000. Daarmee verzekeren we ons ervan, dat de derde groep van opslagadressen daar helemaal bovenin RAM op veilige afstand blijft van de Basic-opslag. Want dat is de Stack met als standaard startadres &HF098. Daarin worden tijdens de uitvoering van ons programma tijdelijk registerwaarden opgeslagen en weer vandaan gehaald. Die stack-adressen zijn het gewend omlaag te groeien. Ze komen de Basic-opslag-adressen tegemoet. Gebruiken we al te veel adressen, dan riskeren we de foutmelding: Stack overflow. Maar niet doen dus.

Het systeemgebied vanaf adres &HF390 tot het allerhoogste adres &HFFFF is een verhaal op zich. We weten inmiddels al, dat hier onder andere die te wijzigen teksten van de funktietoetsen worden opgeslagen. Tikt u adres 62441 in dan is vast te stellen, dat dit met de aan COLOR verleende waarden eveneens het geval is. En als we dan adres 63178 ingeven, dan komen we cijfer 8 tegen. Nu niet omdat onze computer een voorkeur voor die 8 heeft.
Hier staat deze 8 voor het data-type volgens het veronderstelde in ROM. We hebben in het Peek-programma namelijk dat data-type niet gedefinieerd middels bijvoorbeeld DEFSTR. Zouden we dat gedaan hebben, dan had er in deze adressen cijfer 4 gestaan. Zo staat hier de 2 voor een integer en de 4 voor enkele precisie. Zonder definitie wordt de 8 voor dubbele precisie aangenomen. Indien het Basic-programma zich daarvoor leent, als dus maar één type variabelen wordt gebruikt, dan is het een goede zaak om dit vooraf te definiëren. Als dubbele precisie niet nodig is, dan zullen we het evenmin door ROM laten veronderstellen. Voor de uitvoering worden gewoon meer bytes gebruikt. Worden desondanks andersoortige variabelen gebruikt, en dat verhindert zo'n definitie niet, dan hadden we het beter achterwege kunnen laten.

In de ROM-routines kunnen we hier en daar CALL's tegenkomen,

waarvan de adressen in het systeemgebied te zoeken zijn. Er zit er eentje in adres 2240. Hierbij hoort adres &HFDA4 (dec.64932). Aldus vol verwachting dit adres ingetypt. Of nog beter adres 64922 en dan nog even doorgaan tot adres 65481. Het is niets dan RET en nog eens RET wat we daar tegenkomen. Zo heeft het er de schijn van, dat zo'n CALL die lange reis moet maken met geen ander doel, dan bij zo'n RET meteen weer rechtsomkeerd te maken. Zo is het ook, zolang wij niet met deze RET-adressen gaan rommelen. Een CALL heeft maar 1 RET nodig, maar voor elke CALL zijn er 5. Bij elkaar 560 keer RET in plukjes van 5 over 112 adressen. Als u het echt zo precies wilt weten. Dat er 5 RET's zijn, biedt de kans om in 4 daarvan vanuit bijvoorbeeld machinekode andere waarden en/of bewerkingen te zetten, die de uitvoering van de betreffende ROM-routine kunnen beïnvloeden. Het dient vanzelfsprekend wél met de nodige deskundigheid te worden voltrokken. Ze zijn niet gek in ROM. We hebben het over HOOKS en die beperken zich niet tot deze RET-adressen. Het is er natuurlijk niet voor de grap. Het ontwerp van de MSX is er op gericht aanpasbaar te zijn voor ook niet typische MSX-software. Wanneer in zo'n programma nu maar ingespeeld wordt op dit aanpassingsvermogen, dan kan zo'n brok software heel andere zaken tot stand brengen, dan gebruikelijk is. Vanuit Basic kunnen we ook het één en ander naar onze hand zetten. Met: POKE &HFF89,225 schakelen we bijvoorbeeld de LIST-funktie uit. In het geval u ooit een supergeheim programma ineen knutselt, dan zou u zo'n POKE kunnen opnemen. Te LISTen is het dan niet meer. Ook niet door u! Dat levert alleen "Syntax error" op. U kunt het even uitproberen bij het PEEK-programma. Het vervelende mag hier zijn, dat met POKE &HFF89,0 het LISTen weer mogelijk wordt. En dát weet natuurlijk ook de eventuele kraker van uw programma. Ik heb er nog eentje voor u straks. Van tijd tot tijd worden deze hooks gepubliceerd in de computerbladen.
Goed om te verzamelen. Zelf experimenteren kan ook. U moet zelf maar uitmaken in hoeverre zoiets ten koste van eigen Basic-programma's mag gaan.

Vanzelfsprekend blijft het PEEK-programma paraat voor het geval u nog eens op uw gemak langs die adressen wilt wandelen. U kunt het programma bijvoorbeeld opnemen in uw Basic-programma. Dan is vooral in VRAM te kontroleren, wat u er hebt aangericht door bijvoorbeeld sprites te ontwerpen. Onze reis was een loutere oriënterende. Daarmee hebben we gevolg gegeven aan de kanttekening die bij VPOKE wordt gezet, namelijk dat we inzicht nodig hebben omtrent de geheugenopbouw. Wat we er nu van weten, is meer dan toereikend.
Niet, alsof er nu alles van bekend zou zijn. Bij lange na niet. Daarvoor zijn andere boeken te raadplegen. Dat wordt pas echt van belang, als

we ons met machinekode willen bezig houden. Veel beter is evenwel, om eerst fatsoenlijk in Basic te kunnen programmeren en dat bewezen te hebben in eigen programma's. Van de hak op de tak springen is bij het computeren ook al niet aan te bevelen. Een langzame doch gedegen uitbouw van onze programmeervaardigheden levert tenslotte de gezochte voldoening op. Dan die laatste hook. Type POKE &HFF07, &HC7 en geef RETURN. Ja, een komplete reset. Programma weg, scherm gewist en startscherm terug. Dit kunt u niet ongedaan maken! De Ram-adressen waarin de Basic-regels waren opgeslagen zijn geveegd. Zo willen we het ook. Want we moeten langzamerhand onze trip langs het elektronische afsluiten. Alleen wanneer er óók machinekode in RAM zou zijn opgeslagen, dan zou dat niet zijn gewist. Om zulke bytes te ontslaan, zou de machine uitgezet moeten worden en na enige sekonden weer aan.

Besluiten we aldus met een leesoefening van RAM-adressen, waarin het volgende Basic-programma zal worden opgeslagen. Alweer Peek!

```
10  CLS
20  PRINT "MSX"
30  CLS:FOR A=32768! TO 32786!
40  PRINT A;" "; PEEK (A)
50  NEXT A
RUN
```

Dat is nog wel in te typen, nietwaar? De adressen in de lus zijn die, waarin regel 10 en 20 zijn opgeslagen. Om het beknopt te houden. U kunt uiteraard zelf de adressen van de overige regels nalopen en op dezelfde manier lezen, als we ons hier voornemen te doen. Op het scherm behoort het volgende te staan met hieronder een verklaring bijgevoegd.

```
32768  0     ' startadres RAM
32769  7     ' met volgende regel low/highbyte van adres regelnum-
32770  128   ' mer 20 (=32775)
32771  10    ' regelnummer 10
32772  0     ' gebruikt bij regelnummers hoger dan 255
32773  159   ' tokennummer van CLS
32774  0     ' einde Basic-regel 10
32775  19    ' met volgende regel low/highbyte van adres regelnum-
32776  128     mer 30 (=32787)
32777  20    ' regelnummer 20
32778  0     ' voor regelnummers hoger dan 255
32779  145   ' tokennummer van PRINT
```

32780	32	' ASCII-code voor spatie
32781	34	' ASCII-code voor "
32782	77	' ASCII-code voor M
32783	83	' ASCII-code voor S
32784	88	' ASCII-code voor X
32785	34	' ASCII-code voor "
32786	0	' einde Basic-regel 20

Die tokenlijst in APPENDIX-IV is ervoor bedoeld om dit lezen van de Basicopslag ook voor andere instrukties mogelijk te maken. Wanneer u hieraan althans behoefte mocht hebben. Enige oefening kan echter nooit kwaad. Waar in een Basic-regel een getal wordt gebruikt, daar geven de adressen codenummers welke daarvoor staan. U kunt desgewenst nog even vervolgen met de overige regels van dit Basic-programma door van TO 32786! hiervoor TO 32835! te maken. En als u zelf dat uitroepteken niet achter het getal zet, dan doen die zakelijke burgers van ROM dat wel. Het staat voor enkele precisie.

Nuttig is dit lezen van de Basic-opslag alleen om er het geheimzinnige van weg te nemen. U ziet evenwel bij adres 32780 óók, dat een spatie als volwaardig wordt beschouwd. Het kost 8 bytes per gebruik. Basic-regels hebben geen spaties nodig, behalve als dit voor een instruktie is aangegeven. Desondanks moet u die spaties althans voorlopig maar royaal blijven gebruiken. Evenzeer is het verstandig, om meteen niet al teveel instrukties in één regel te persen. Later bij lange programma's of als veel bytes voor de uitvoering nodig zijn, kunnen we alles aan elkaar typen met zo vol mogelijke regels. Dat 'eerst duidelijk, dán snel' zullen we voorlopig nog even in ere houden. Trouwens, als een programma na het RUNnen nog heel wat bytes ongebruikt laat, zoals ten alle tijde via ?FRE(0) is vast te stellen, dan zal duidelijkheid het winnen van spaarzaamheid.
Wat kan er de zin van zijn om dan nóg meer bytes vrij te houden door bijvoorbeeld alle spaties uit de weg te ruimen? Het is meegenomen, dat spaties achterwege kunnen blijven en regels heel lang kunnen zijn, maar we moeten wél het overzicht behouden. Het moet zonder tranende ogen leesbaar blijven. Zéker zolang nog aan een programma wordt gespijkerd. Echter ook naderhand is de leesbaarheid meegenomen, als we in een listing nog eens willen nagaan hoe we dit of dat voor elkaar hebben weten te krijgen. De computer laat bij bespaarde bytes echt niet weten: Thank you very much, fellow! Zoiets betreft ook de REM-regels. Dat REMarken is heel handig. Bytes vergt het wel. Als we daarvan toch genoeg hebben, dan zullen we ze gebruiken ook. Anders wordt het naderhand erg zoekerig, als we willen nagaan waar ergens een subroutine is ondergebracht. Bij de opbouw van een pro-

gramma moeten we gewoon kwistig REM-regels gebruiken om te blijven beseffen waarmee we bezig zijn. Loopt de boel eenmaal naar wens, dan kunnen we de niet echt nodige REM-regels altijd nog verwijderen. Listings in computerbladen met volgestouwde regels, zonder spaties en met amper een REM-regel zullen ons niet ten voorbeeld zijn. Daarvoor moet echt een zinnige reden bestaan. We hebben het er nog wel eens over bij het opzetten van een programma. Het laatste onderdeel. En al zijn we hier maar net begonnen, we blijven nog even in deze sferen.

hoofdstuk 4
Kleur en tekst

Mocht u even niets om handen hebben, dan zou u programma nr.3 LETTER vanaf de listing kunnen intikken. Het is een gekombineerd programma, waarvan er meer zullen volgen. Allemaal korte programma's intypen is ook niet het ware. Zo kunt u tenminste even vooruit met het in praktijk brengen van uw typevaardigheid.
Aldus eerst nog even in het sfeertje van het programmeren. We hadden het al even over de overbodigheid van KEY LIST. Met die funktietoetsteksten (een prima Scrabblewoord!) is nog iets aan de hand. Het kan erg vervelend zijn bij een vol scherm, waarvan we alle regels moeten benutten. Nu, dat lukt niet bij screen0 en screen1. Toch zullen we het laatste scherm het meest gebruiken. De onderste schermregel 23 is kennelijk gereserveerd voor die teksten. Om dit voor uw machine te kontroleren, je kunt het maar beter zeker weten, zult u na RUN eerst even toets 1 indrukken. Het MENU nodigt uit om diverse cijfertoetsen een duw te geven bij het afwisselen. Via toets 1 en toets 2 ziet u de regels weergegeven tot en met nummer 23. Als nu ook bij u de bovenste REGEL: 0 door het scrollen verdwijnt, dan mag u het PRINTen van regel 23 vergeten. Dat zou in een scroll resulteren en niet overeenkomstig de bedoeling zijn. Daarmee dient rekening te worden gehouden. Omdat screen2 via een file tekst binnenhaalt, hebben we er daar geen last van. Maar al te goed bruikbaar is de tekst hier niet. Ondanks KEY OFF blijft regel 23 voor die teksten gereserveerd. Zondermeer een misser(tje) van de ontwerpers. Bij KEY OFF had regel 23 vrij dienen te komen voor PRINTgebruik. En je kunt hem hard nodig hebben die laatste regel!

Niets is evenwel helemaal onmogelijk met MSX. Daarmee introduceren we eindelijk, waarom het hier allemaal gaat. Het gebruik van VPOKE en VPEEK. Want drukt u toets 4 of 5 in dan ziet u regel 23 wél van karakters voorzien. Gemakshalve zijn er de karakters 65 t/m 104 voor genomen. LISTen we het programma, dan komen we in regel 980 de VPOKE tegen via welke we die regel toch kunnen benutten. We doen dat, door de kodes direkt te VPOKEn in VRAM-adressen, waar men zich met de schermregistratie bezig houdt.

```
980  D=BASE (5) +736     'vanaf 736 begint regel 23
990  FOR I=0 TO 31       'om 32 karakters weer te geven
1000 A=A+1
1010 VPOKE D+I, A
1020 NEXT I
```

In regel 950 krijgt A een beginwaarde van 64. Zo wordt 65 als het kodenummer van de letter A, het eerste te VPOKEn karakter. De zo aan de variabele A toevertrouwde kodes worden vanaf schermlokatie 736 achtereenvolgens op het scherm gezet. Nu is het uiteraard erg gemakkelijk om hier met een variabele te kunnen werken. Willen we echt tekst op regel 23 hebben, dan wordt het wat ingewikkelder. Die adressen van BASE (5) werken met karakterkodes. Voor tekst moet aldus de kode van elke letter uit een DATA-regel gelezen worden met uiteraard READ in die lus. En het moet decimaal. Het kan heel goed, maar het mag niet al te warm worden aanbevolen. Eerder is deze methode te gebruiken, om bij het gebruik van een kleurbalk onderaan het scherm daarmee ook regel 23 te voorzien. Dan komt tekst in zo'n balk niet helemaal onderaan te staan. Zoals u kunt zien, is vooraf in de regels 960 en 970 een andere kleur gegeven aan de af te drukken letters. Het is een eenvoudige bezigheid om een karakterset of meerdere een eigen kleur te geven. Wanneer u in het hoofdmenu nu toets 2 indrukt, dan mag u nog eens toets 1 indrukken om wat in beeld te krijgen. We willen het eerst even hebben over die verschillend gekleurde vlakken op het scherm. Ja, het is screen1. De LINE-instruktie is hier niet te gebruiken. Hoe krijgen we toch van die vlakken? Of balken of wat dan ook en waar dan ook op dit zo gebruiksvriendelijke scherm? Zonder VPOKE in elk geval niet. Dat hebt u ongetwijfeld al lang zelf ontdekt. Gesteld nu dat we een donkerrood vlak ergens op het scherm willen, dan zoeken we eerst een MSX-set uit, waaraan we deze kleur willen geven. Nemen we set 17.

```
A=BASE (6)
```

Al zijn er meerdere karaktersets in verschillende kleurenkombinaties te VPOKEn, het is afdoende om in de beginregel éénmaal de BASE-waarde aan een variabele toe te vertrouwen. Ofschoon daarvoor natuurlijk elke letter gebruikt kan worden, is het terwille van de herkenning wijs om voor regelmatig terugkerende zaken dezelfde variabele te bezigen. Dat is óók een manier van standaardiseren.

```
FOR C=128 TO 135
```

Van het eerste tot het laatste kodenummer van set 17. Het is niet zo

zinnig om hier naar willekeur kodes te kiezen. Dan kunnen karakters gekleurd worden, welke we dat genoegen niet hadden gegund. Vandaar ook die verdeling in 32 sets in APPENDIX-I. Zo is begin en slot kode van een set meteen te onderscheiden. Ik gebruik doorgaans de variabele C voor Codes. Ook als C$ bij READ. Altijd te herkennen zo.

VPOKE A+ (C/8), &H66

Hier draait het helemaal om. Dat delen door 8 houdt verband met de opbouw van een karakter uit 8 bytes. Die willen we ook allemaal de gewenste kleur geven. Dat kan toch al niet anders. Alleen bij screen3 kan elk van die 8 een eigen kleur krijgen. Een andere manier van VPOKEn van kleur is kennelijk niet toegestaan. Immers, bij die reis door VRAM zagen we, dat voor elke set een eigen kleuradres bestaat. Het funktioneert echter niet, indien we direkt in zo'n adres VPOKEn. Zoals het er nu staat, wordt bij elke doorloop het startadres waarvoor BASE(5) staat, verhoogt met de gedeelde C-waarde. En dat werkt prima. Houden zo.

In plaats van kleurkode &H66 kan ook decimaal 102 worden gebruikt. In &H66 evenwel herkennen we meteen, dat het om de kombinatie donkerrood (kleurkode 6) op donkerrood gaat. Alweer een reden om aan dat hexadecimale vast te houden. Zolang u het maar beperkt tot het computeren. Een wet van Meden en Perzen zal dat niet zijn. De decimale kode is weer heel geschikt om aan een variabele toe te kennen. Dat er met NEXT C besloten wordt, had ik niet behoeven te schrijven. Deze methode om een gekleurd blokje te verkrijgen, waarmee zo'n vlak te PRINTen is, kent als nadeel, dat daardoor álle karakters van set 17 die metamorfose hebben ondergaan. Terwijl we er slechts één nodig hebben. Zitten we royaal in de sets, dan kiezen we er toch voor. Moet er zuinig mee worden omgesprongen, dan verdient een andere methode de voorkeur. Nemen we karakter 128 als de eerste van set 17, dan moet deze hiervoor eerst veranderd worden in een vol blokje. Aldus in een 8x8 matrix met als patroonwaarden 8x &HFF of 255. Als kleurkode gebruiken we dan &H60. Dan blijft de achtergrond transparant en kunnen de overige karakters gewoon op het scherm komen. In rood. Ze zijn nog te gebruiken om zelf ontworpen figuren in te VPOKEn. Hoe transparant evenwel ook de achtergrond van bijvoorbeeld letters ook mag zijn, ze zullen toch niet te plaatsen zijn in zo'n kleurvlak. Die achtergrond past zich altijd aan bij de in COLOR bepaalde schermkleur. Zo meteen meer hier over.

Via welke methode ook, we hebben karakter 128 als de ç ter beschikking voor dat kleur vlak. Want met PRINT "ç" zal nu een donkerrood

blokje op het scherm komen Voor een enkele balk herhalen we die ç gewoon net zo vaak in PRINT als het ding lang moet worden. Voor een vlak gaan we natuurlijk in elk geval wat efficiënter te werk. Stel dat vanaf kolom 10 en regel 5 tot aan kolom 20 en regel 10 dat kleurvlak moet komen. Dan gebruiken we het gemak van de lussen.

```
FOR X=10 TO 20
FOR Y=5 TO 10
LOCATE X,Y:PRINT "ç"
NEXT Y,X
```

Het kan allemaal in één regel staan. Zo is overal op het scherm een kleurvlak neer te planten of een reeds omlijnd kader een kleur te geven. In het laatste geval moeten we natuurlijk ook de kaderlijntjes van set 3 en 4 VPOKEn met als achtergrondkleur donkerrood. Het nadeel van deze bestaande lijntjes is, dat zij in het midden van een 8x8 matrix zijn gezet. Hierdoor zal de gegeven achtergrondkleur ook aan de buitenzijde te zien zijn. Als we dat niet willen, dan moeten zelf kaderlijntjes worden ontworpen en die ontwerpjes in bijvoorbeeld deze sets geVPOKEd worden. Zeker, met LINE, DRAW en/of PAINT besparen we ons al die moeite. Dán hebben we echter wel met het speciaal grafische scherm van screen2 te maken. Dan kunt u niet aan kleurwijziging doen van karaktersets en al evenmin een karakter veranderen in een eigen ontwerp. Ook zal het daar niet lukken om bijvoorbeeld een strak, zwart lijntje om een rood vlak te zetten. Zo is screen0 weer het typische tekst-scherm. Daar is kleurwijziging evenmin mogelijk. Alles dus in de eigen waarde laten. U zult uw computer ook niet bij de groenteman hebben gekocht. Wat wellicht óók helderder zal gaan schijnen, kan de gedachte zijn, dat het werkelijk geen puzzel mag zijn om een willekeurig karakter op het scherm te zetten. Niet voor de tekstverwerker en zeker niet voor de VPOKEr in screen1.

Het tweede aspekt hier betreft het plaatsen van tekst op een kleurvlak of in een kleurbalk. Doen we dat zondermeer, dan ontstaat het zeker maar al te vertrouwde beeld, zoals ons wordt getoond in het magentavlak. De achtergrond neemt de schermkleur aan.
Niet zo'n fraai gezicht. Een klacht inzake MSX is soms, dat de karakters slechts per set van 8 anders gekleurd kunnen worden. Dat is echter niet typisch MSX. Het betreft alle computers. Wanneer het tòch mogelijk is, om bijvoorbeeld een enkele letter een aparte kleur te geven, dan bedient men zich van een truuk. En dat kunnen wij net zo goed. Wanneer u achtereenvolgens toets 1, 2, 3 of 4 indrukt, dan ziet u daarvan het harde bewijs. Steeds dezelfde letters maar telkens in een andere kleur met de achtergrondkleur aangepast bij die van het kleurvlak.

Het kan dus zeer wel. De oplossing zit andermaal in dat zo ten onrechte als een stiefkindje behandelde VPOKEn. Wat we voor een tekst nodig hebben aan letters, cijfers of leestekens, dat VPOKEn we simpelweg in andere karakters. Zoals die er zitten vanaf bijvoorbeeld set 17. Aangezien we deze set al gebruikten voor dat rode blokje, beginnen we welgemoed aan set 18.

```
A=BASE (7) + 384:B=BASE (7) + 1088
```

Maar even anders dan bij het kleur VPOKEn. Aan A wordt het startadres van de karaktersets van screen1 toegewezen We wensen evenwel te beginnen bij het adres, waar de patronen van de cijfers 0 t/m 7 zijn opgeborgen. Daarvoor hoeven we dat PEEK-programma niet weer van stal te halen. Het eerste te gebruiken kodenummer is 48; dat van cijfer nul. Omdat een karakter uit 8 bytes of adressen bestaat, vermenigvuldigen we die 8 even met 48 en verkrijgen 384. Hadden we dat lachebekje van kode 1 nodig gehad, dan was het nog vlotter gegaan door 1x8=8. Begrijpt u wel? Door die 384 toe te voegen aan variabele A, is het het startadres geworden van de karakters die we willen overhevelen naar soortgenoten in set 18. In die set is kode 136 als eerste geplaatst. Weer even vermenigvuldigen met 8 en 1088 zal het resultaat zijn. Daarmee hebben we het tweede startadres kunnen bepalen zoals aan de B toegewezen. Het is helemaal niet nodig om de acht patroonwaarden van elk van die acht cijfers via READ en DATA-regel in de adressen van set 18 te stoppen. Dat doen we uitsluitend, als het om eigen ontwerpen gaat. Het ontwerp van die cijfers zit al kant en klaar in de adressen.

```
FOR I=0 TO 63
C=VPEEK (A+I)
```

In de cijferset zitten 8 karakters á 8 bytes en zo zijn er aldus 64 te VPEEKen of te lezen uit die adressen. Door A+I klauteren we steeds een adres hoger. Het moet wel tussen haken staan. En aan C worden deze patroonwaarden achtereenvolgens toegekend.

```
VPOKE B+I, C
NEXT I
```

Wat aldus middels VPEEK wordt gelezen, dat wordt via VPOKE weggeschreven in de adressen van set 18. Zoals die starten met de variabeleB. Daarmee is het drama voltrokken, dat de karakters 136 t/m 143 veranderden in de cijfers 0 t/m 7.
Typen we nu karakter 137 op het scherm, dan zien we in plaats van

die ë cijfer 1 verschijnen. Een kind zou de was kunnen doen. De luswaarde als I gegeven, is hier de tijdelijk toegevoegde waarde aan B. Als we aansluitend de karakters van de volgende set eveneens willen wijzigen, dan kan, zonder weer iets bij B te hoeven tellen, in de lus van B een teller worden gemaakt.

```
VPOKE B,C
B=B+1
NEXT I
```

Na de lus zal B zo met 63 zijn toegenomen en is weer startadres voor set 19. VPOKEn we eigen ontwerpen in karakteradressen, dan kunnen we daar soms heel wat sets voor nodig hebben. Het enige dat dan te doen is, is het aantal te gebruiken karakters weer met 8 vermenigvuldigen en het resultaat als luswaarde te gebruiken. Omdat we die altijd met 0 starten –in de eerste doorgang behoeft niets te worden bijgeteld– wordt zo'n resultaat uiteraard met 1 verminderd.

Daarmee zijn we er wat onze cijfers betreft nog niet helemaal. We willen die dingen immers op dat rode kleurvlak PRINTen. Laten we het houden op lichtgeel

```
A=BASE (6)
FOR C=136 TO 143
VPOKE A+ (C/8), &HB6
NEXT C
```

Omdat BASE (6) al gebruikt is bij dat rode blokje, had het hier niet nog eens behoeven te worden gebruikt. Voor een beginnende VPOKEr kan het echter geen kwaad. Nu zult u allicht zeggen, dat &HB6 ook niet meteen duidelijk maakt, dat het om lichtgeel op donkerrood gaat. Een kwestie van wennen. Die B is de tweede vervangende letter voor getal 11 en dat is dus lichtgeel. Decimaal is het 182. Had u daar eerder de kleurenkombinatie in herkend? Nou dan!

Willen we nu op het rode vlak het getal 135 gePRINT hebben, dan is dit thans zonder wiseffekt mogelijk door PRINT "ëïì". Ja, omgeturnd in 135. Het wordt alleen wat gekompliceerder, als dit getal voortdurend moet veranderen. Bijvoorbeeld omdat het een score moet voorstellen. Dan hebben we de IF . . . THEN voorwaarde-regels nodig. Dat vermijden we bij voorkeur en laten ergens anders scoren. Voor zo'n aangepaste tekst op kleurvlakken is het nog het beste een vaste tekst te kiezen. Dan kan voordien gekozen worden tussen hoofdletters of kleine letters. Ofwel tussen kapitalen en onderkast, zoals van oudsher aange-

duid in de dagen van de letterbak.
Dan is bijvoorbeeld de hele set kleine letters achter elkaar te VPOKEn in andere karakters. Het kan evengoed voor één enkele letter worden gedaan. Wie in een programma bijvoorbeeld de aandacht wil vestigen op een mededeling, die zou daarvan de eerste letter een andere kleur kunnen geven. Wat u maar wilt. Hier zijn die cijfers alleen geVPOKEd, om ze elders op het scherm in hun bij COLOR bepaalde kleur ook te kunnen PRINTen. Zou dat niet nodig zijn, dan behoeven we alleen die cijferset de juiste kleur te geven om ermee op een kleurvlak te kunnen PRINTen. In de praktijk zal dit nog het meest voorkomen. Die andere manier dienen we echter ook te kennen.

In het programma heb ik het me wat moeilijker moeten maken. Dezelfde letters in diverse kleurenkombinaties betekende, dat deze letters óók in even zovele sets gestopt moesten worden. Die dan eveneens de gewenste kleur dienen te krijgen. In een programma kan altijd maar het beste begonnen worden met zo'n kleurverandering. Daarna de verandering van karakters in wellicht eigen ontwerpjes en tenslotte eventueel het direkte plaatsen daarvan op het scherm in BASE(5)-adressen.
Wanneer u toets 5 indrukt, dan brengt dit een INPUT op het scherm. U ziet er ook enige zinnetjes samengesteld uit de gebruikte karakters. Door nu een (decimale!) kleurkode in te geven is de letterkleur te veranderen. De opgesomde kleurkodes hebben allen zwart als de schermkleur als achtergrondkleur. Dit kan een goede gelegenheid voor u zijn om kleurkontrasten te beoordelen door ook andere kodes in te voeren. Ze staan allemaal in APPENDIX-III. Daar is ook een kontrastwaardering genoemd, ofschoon het mede afhangt van de gekozen schermkleur. Bij het opzetten van een schermbeeld, eerst nog op papier is het wel zo gemakkelijk om te weten welke kleurenkombinatie het het beste zal doen. In de lijsten is uitgegaan van de schermkleur 4. Het zou eigenlijk in de handleiding behoren te staan. Dat is, voorzover bekend, bij MSX nooit het geval. Enig begrip is daarvoor wel op te brengen, gezien alle toelichting die nodig is bij de vele mogelijkheden van de computer. Zo moeten we ons ook deze basis-informatie zelf verschaffen. Want we moeten gewoon vlot kunnen werken. Ook met de kleuren. Het goede gebruik van de Basic-instrukties kost al genoeg hoofdbrekens. De tweede INPUT nodigt u uit met de aangegeven kodes zelf woordjes samen te stellen. Als u dat te veel van het goede is, dan drukt u gewoon op RETURN of typt u het alfateken @ om naar het hoofdmenu terug te keren. Naderhand zullen we dieper ingaan op die kleurenkombinaties. Het gebruik van VPOKE en VPEEK moet zodanig vertrouwd worden, dat het altijd zondermeer kan worden toegepast. Daarom zullen we nog wel eens in herhalingen vervallen. Tenslotte: mocht u beschadigde letters op het scherm hebben gezien, dan komt dit door het gebruik

van screen2 in het programma. Schakel de computer even uit. We willen toch al het volgende programma laden.

hoofdstuk 5
Het vierde net en de kleuren

Het gebruik van screen 1 voor verreweg de meeste programma's steunt geheel op de ervaringen van vergevorderde programmeurs. Daarmee is het prettigste te werken met een breed scala aan mogelijkheden. Tekstverwerking voor screen0 en fraaie grafische voorstellingen voor screen2. Er kan evenwel niets tegen zijn, om dit grafische scherm voor een titelbeeld van ons programma te gebruiken. Dan liever geen tekst gebruiken om het risiko van beschadigde letters bij de overgang naar screen1 te vermijden. Screen3 is als ons vierde net het muurbloempje onder de screens. Wat moeten we er eigenlijk mee? Al te gek veel is er inderdaad niet mee te doen. Het is een typisch kleurenscherm, waarop wél heel fraaie beelden tevoorschijn kunnen komen. Zo kan het als titelscherm goed bruikbaar zijn. Hoe fraai echter kleurplaatjes ook mogen zijn, we willen toch vooral een praktisch gebruik maken van de computer. Al is het ook voor een spel. Ik mag u opwekken programma nr. 4 COLOR3 in te typen. U kunt er even mee vooruit. Het gaat hier echter om screen3 en allicht zal het u alle moeite waard kunnen zijn. Intussen doe ik net, alsof het allemaal sekondenwerk is.

Na de RUN gebeurt er niet al te veel. MENU is maar eens weggelaten deze keer. Screen3 wordt goeddeels gebruikt en steeds zo'n overgang naar screen1 is ook niets gedaan. Het programma zelf vergt minder dan 7000 bytes. Als u na de uitvoering weer ?FRE (0) geeft, dan zal blijken, dat het allemaal een 20.000 bytes heeft gevergd. Screen3 weet van innemen! In het programma zitten 6 mogelijkheden, bereikbaar via de cijfertoetsen 1 t/m 6. De 6 is END. Na afloop van elk onderdeel hoort u een toontje ten teken dat de volgende toets ingedrukt kan worden. Beginnen we ordelijk met toets 1. De grafische instrukties van screen2 kunnen gebruikt worden. Aan het magenta PSET-stipje is af te lezen, dat alles viermaal zo groot wordt. Men noemt het "low resolution". Dat geldt ook voor de tekst, zoals cijfer 3 hier afdoende duidelijk maakt. Voor slechtzienden allicht een uitkomst. Sprites kunnen dus ook en zonder formaatvergroting groeien ze toch wel buiten de gegeven proporties; het behoort een 8x8 sprite te zijn. DRAW, LINE en ook PAINT zijn bruikbaar, als we met deze uitvergroting rekening willen houden. CIRCLE leent zich niet zo goed. We moeten

wél speciale motieven hebben om met deze instrukties gebruik te maken van screen3. Maar het kan en wellicht raakt u zo gemotiveerd.

Wat na het indrukken van toets 2 komt, zal wat interessanter kunnen zijn. De presentatie van alle 256 kleurenkombinaties. Hier weliswaar speciaal bedoeld voor screen3, maar zij gelden natuurlijk evenzeer voor ons echte werkscherm screen1. Hier is overigens van screen2 gebruik gemaakt. Laat u aldus niet misleiden door die drie.
Van de weergegeven kleurkodes treft u de decimale en hexadecimale dus aan in APPENDIX-III. Het gaat hier echter speciaal om screen3. Het kent een afwijkende opbouw. Wat in hygiënisch blauw onder A en de B is voorgesteld op het scherm, wil een voorstelling van deze opbouw zijn. Een blokje bestaande uit de bekende 8 lagen waarin onder A de voorgrondkleur en onder B de achtergrondkleur is te plaatsen. En elke laag kan in een andere kleurenkombinatie worden gezet. Op die manier kan op het scherm een bonte verzameling kleurblokjes ontstaan, die wij eventueel tot een bepaalde figuur kunnen proportioneren. Dat is wel de belangrijkste gebruikswaarde van dit scherm. Het bouwt zich van boven naar beneden op. Zodra laag 8 gekleurd is, wordt aan laag 1 van het volgende blokje begonnen. Zo gaan er 32 van zulke 8-voudig te kleuren blokjes op een regel. We hebben 6 van deze regels en aldus kunnen er 1536 gekleurde stripjes of het dubbele daarvan aan blokjes worden weergegeven. Ja, we kunnen kleurige ontwerpen maken voor ons plezier of om als titelscherm te gebruiken. Maar dat is niet niks. Eenvoudiger maken we het ons, door alle kleurkodes via RND (1) naar willekeur op het scherm te brengen. Het zo steeds wisselende resultaat is te bewonderen, wanneer toets 3 wordt ingedrukt. De genoemde schermopbouw is goed te zien. Fraai is het zeker. Het kan er een idee van geven hoe een kleurtekening er zou kunnen uitzien. We houden het wat beter in de hand, wanneer we gebruik maken van de COS of SIN instruktie van Basic. Dat is weer te bewonderen na het indrukken van toets 4. Het vergt enige tijd. Al tovert het een fraai beeld op het scherm, al te veel is er niet mee te doen. Het kan hooguit een ideetje zijn voor weer zo'n titelbeeld, waarmee we ons programma kleurrijk willen introduceren. En waarom zouden we niet? Het oog wil ook wat.

Wat met kleur mogelijk is in MSX, dat staat alleen screen3 ons toe. Een voorbeeld van zo'n titelscherm is ineen gezet. Toets 5 is hiervoor in te drukken. We rennen door dit programma heen. Een prijs in een mozaikwedstrijd zal er niet mee te winnen zijn. Het vraagt een vracht aan DATA-regels, waarin al die kleurkodes zijn onder te brengen. PRESET en PRINT zijn hiervoor niet te gebruiken. Wél het VPOKEn in de schermlokaties van screen3 en dat is dan ook gedaan. Nu is er wél een tekenprogramma te maken, waarmee middels INPUT een

kleurkode op de gewenste plek kan worden gezet. Daarbij kan van een aanwijssprite gebruik worden gemaakt. Rijen gekleurde blokjes kunnen de kleurplaatsen markeren. Dán kunnen wel heel fraaie kleurtekeningen worden gemaakt. Lokaties en kleurkodes zijn in een file onder te brengen, zodat ze naderhand weer voor zo'n tekening boven water zijn te halen. U zou er de tanden eens in kunnen zetten. In zo'n tekenprogramma wordt hier bedoeld.
Het voornaamste hier waren die kleurenkombinaties. Want daarmee valt er zo het één en ander te doen in screen1, zoals we al hebben kunnen zien. Wijziging van karakters is overigens bij screen3 evenmin mogelijk als bij screen2. Nog een laatste opmerking in deze "output". In onder andere regel 140 is gebruik gemaakt van LINE/BF om het scherm de gewenste kleur te geven. Want al is deze in COLOR aangegeven, wisseling van schermkleur in een programma heeft niet altijd direkt het gewenste effekt. Pas bij een tweede RUN wordt dan die schermkleur aangenomen. Om dit te vermijden is dus gebruik gemaakt van LINE/BF. Algemeen zal het niet zo vaak gebeuren, dat de schermkleur moet veranderen. En bij screen1 geeft het doorgaans niet zo veel problemen. En zo toch, dan levert LINE ons hier slechts een foutmelding op.

hoofdstuk 6
Schermlezing

Tijdens dat doorlopen van VRAM zijn we BASE (5) van screen1 al tegen gekomen. De opbergadressen van wat er aan karakterkodes op het scherm kan staan. Het terechte vermoeden was reeds bij u opgekomen, dat we daarmee heel aardige dingen kunnen doen in een programma. Schermlezing via VPEEK is zelfs eenvoudig onmisbaar in programma's met schermkontrole. Het is ook toe te passen bij screen0. Daar zal het zich echter beperken tot die zogenaamde ikoontjes bij het gebruik van een tekstverwerker. Want al zijn zulke professionele programma's doorgaans in machinekode geschreven, vanuit Basic kunnen wij precies dezelfde methode van aanwijzing toepassen. Wilt u de linker kantlijn verplaatsen? Ga naar ikoontje X, druk een toets in en ziedaar! Dat ikoontje heeft ook hier geen eigen betekenis. Het gaat alleen om de karakterkode welke hiermee is verbonden. Zou dat karakter 128 zijn, dan is het deze kode welke van het scherm gelezen wordt en waarmee de routine is verbonden voor die kantlijnverplaatsing. En het maakt niets uit, waar ergens op het scherm dat ikoontje staat. De aanwijssprite markeert elke schermlokatie en alleen die plek wordt gelezen. Zo kunnen ook wij met screen1 een programma aldus heel professioneel maken door eveneens met zulke ikoontjes te werken. Niets geen toetsen indrukken. Alleen maar aanwijzen, wat je gedaan wilt hebben. U zult dit nog tegenkomen in de overige programma's. We kunnen er cursortoetsen/spatiebalk of joystick/vuurknop voor gebruiken. Het is weer iets anders dan het neerknallen van space-invaders.

U kunt u daarvan zelf een toch wel aardige demonstratie voorschotelen, wanneer u programma nr.5 CALCUL "even" wilt intypen.
Het is alweer méér dan louter een demonstratie van schermlezing. Het VPOKEn van zowel kleur als eigen ontwerpen in karakters komt er eveneens royaal aanbod. Het kan tonen, wat daarmee zoal mogelijk is. Weliswaar is die programma-naam CALCUL ontstaan uit de eerste 6 letters van CALCULator, maar eigenlijk dekt die benaming de lading heel goed, zoals u weldra zult bemerken. Indien u intussen geRUNd hebt, dan komt weer eens een MENU met 3 mogelijkheden. De derde is END en ik mag adviseren deze voor het laatst te bewaren. Aldus eerst toets 1 ingedrukt voor een demonstratie van deze schermlezing. U ziet

er de sprite langs alle lokaties van screen1 snellen. Het nummer daarvan wordt onderaan het scherm keurig voor u bijgehouden. Met daarnaast de kode welke op zo'n lokatie wordt aangetroffen. Ja, dat is een hele tijd spatiekode 32. Want pas in lokatie 165 zit iets anders. De kode voor de letter C. Als u rap genoeg de STOP-toets indrukt, dan zult u het kodenummer van de karakter waarop de sprite dan staat, onderaan kunnen aflezen. Dat kunt u desgewenst kontroleren aan de hand van de kodenummers met hun karakters in APPENDIX-I. Omdat er eenvoudig niets op het scherm kan staan, dat zonder kodenummer door het MSX-leven gaat, kunnen we dat zonder bovenmenselijke inspanning veranderen in een zelf verzonnen ontwerp. Dat is in veelvoud gedaan, zoals waar te nemen is na het indrukken van toets 2.

In een 16x16 matrix zijn onder andere de toetsen van onze calculatur ontworpen. Daarvoor waren aldus steeds 4 karakters nodig. Vanuit DATA-regels is het allemaal achter elkaar geVPOKEd in de gekozen karaktersets. En zie het resultaat. Het hangt helemaal van uw verbeeldingskracht en creativiteit af, wat er op het scherm zal komen. Het is allemaal gePRINT. De enige sprite in het gezelschap is de aanwijssprite. Die staat op de ESC-toets. O ja, druk even op de spatiebalk. Het venster van de calculator heeft nog dat fabrieksnieuwe en zo wordt ie weer egaal grijs. Met dit PRINTen van in eigen ontwerpjes veranderde karakters hadden er evengoed tien van die toetsen naast elkaar kunnen staan. Kom daar maar eens om bij de sprites. Of probeert u een en en ander eens met DRAW en LINE voor elkaar te krijgen. En als dat zou slagen, dan is het scherm er toch niet mee te lezen. Daarom gaat het hier vooral.

Een gewone calculator behoefde het niet te worden. Zo'n ding wordt veelal te pas en te onpas gebruikt. Ook voor berekeningen die net zo gemakkelijk even uit het hoofd gedaan kunnen worden. Het maakt de hersenen alleen maar lui. Nu, deze calculator pikt zulk misbruik niet in het geval u rekenkundige bewerkingen zou willen uitvoeren.
Het ding reageert dan tamelijk brutaal. Daar komt u snel genoeg achter, als u zo'n berekening laat uitvoeren. Al was het maar 1+1. Wanneer u maar niet verzuimt om tenslotte met de sprite op = te gaan staan en te spatiebalken. Met de cursortoetsen verplaatst u de sprite. Spatiebalken doet u, zodra de sprite op de gewenste toets staat. De invoer-volgorde is net als bij de huis tuin en keuken calculator. Met uitzondering dan van EXP, CHR (het dollarteken kon er niet meer op) en HEX. Pi is alleen te gebruiken, als u eerst een getal aanwijst, dat bijvoorbeeld gedeeld moet worden door pi. En de laatste handeling blijft dus de =-toets. Het resultaat zou u kunnen tegenvallen. Maar het is alleen onzin om cijfers aan te wijzen, al hoeft u dat uw huisgenoot of vriend natuur-

lijk niet te vertellen. Alleen op het bewerkingsteken te gaan staan en dan = is ook genoeg. Na elke "rekenkundige" bewerking wist u de zaak weer door de sprite op ESC te zetten en te spatiebalken. Het is een stukje ongein met geen ander doel dan de kracht van schermlezing te demonstreren. Want met de toets waarop de sprite staat, is simpelweg een karakterkode verbonden (zoals gebruikt om de toets te ontwerpen), die in de voorwaarde-regels aan hier de variabele N een waarde toekent. Door ON N GOSUB wordt dan naar de routines verwezen, waarin de zaken zijn opgeslagen. Het is heel eenvoudig. Maar wat hier nog een grap tracht te zijn, dat kan in uw programma een bijzonder nuttige funktie hebben. Het wordt in nog komende programma's aangetoond. Gebruiksprogramma's.

Te begrijpen mag zijn, dat het toch wel vele werk, nodig om die calculator op het scherm te krijgen, niet helemaal is verzet om een calculator nu eens van een andere zijde te leren kennen. Er is ook een meer nuttige funktie mee verbonden. We hadden het al over dat door elkaar gebruiken van decimale en hexadecimale getallen voor speciaal adressen. Wat via HEX$ is te achterhalen voor een decimale waarde, dat is helaas andersom niet mogelijk. Evengoed gebeurt het regelmatig, dat we de decimale waarde moeten weten van een in hex gesteld adres. Een lus van 0 tot 16535 is geen doen. Daarom zijn in het programma in te voeren hex-getallen gesorteerd volgens het cijfer of de letter waarmee ze beginnen. Hierdoor kon de wachttijd bekort worden tot maximaal 45 sekonden. Evengoed nog tamelijk lang. Het ligt er maar aan hoe hoog de waarde van de laatste hex-getallen is. Is dat bijvoorbeeld 000, dan komt het antwoord meteen en is het FFF, dan is de volle 45 sekonden geduld te oefenen. Er klinkt trouwens een toontje, als de computer de decimale waarde in het venster zet. Als met ESC gewist is, wat nog op het scherm of in het venster stond, dan zet u de sprite eerst op H$. Daarna stelt u het hex-getal samen door achtereenvolgens op de cijfers of lettertoetsen plaats te nemen en te spatiebalken.
U ziet dit hex-getal in het venster verschijnen. Maakt u het groter dan vier getallen, dan komt er een foutmelding en mag u opnieuw beginnen. Staat het gewenste hex-getal er eenmaal, dan weer op = en afwachten maar. Ook als uw getal uit slechts 1, 2, of 3 cijfers/letters bestaat, dan daarvoor géén 0 plaatsen. Het werkt goed al is het zeker niet ideaal. Wilt u er vaker gebruik van maken, dan kunnen de betreffende regels uit het programma worden gehaald en is gewoon met INPUT te werken. Daarvoor moet natuurlijk wel een stringvariabele worden benut. Voor het overige zult u er weinig moeite mee hebben. Ongetwijfeld bestaan er betere methoden voor deze omrekening. Daarvan wordt onder andere gebruik gemaakt in een speciale calculator. Onbekend is me in hoeverre het ding in ons land te koop is. Het schijnt in elk geval tamelijk prij-

zig te zijn.

Ofschoon verwacht wordt, dat u de listings van de programma's op uw gemak zult doorneuzen, hetzij van het scherm of in APPENDIX-VI, moet hier aandacht gevraagd worden voor de regels 390 t/m 430. Het heeft alles met de schermlezing uitstaande. De X/Y waarden van een sprite ontstaan uit de schermindeling in zogenaamde pickels. Dat is ook met screen2 het geval en nu weet u meteen waar ik het over heb. Het formaat gerekend in 256x191 van het scherm. Wij werken evenwel met screen1. In BASE (5) wordt gerekend met de schermbreedte van 0 t/m 31 (wél WIDTH 32 inbouwen!) en de regels 0 t/m 23. Daarom is een omrekening nodig, wil de sprite in de gebruikte waarden de juiste schermlokatie aanwijzen, welke gelezen moet worden. Zo krijgen K(olom) en R(egel) de juiste waarden door die van de sprite eenvoudig door 8 te delen. Zou middels VPEEK steeds het gehele scherm moeten worden afgezocht naar kodes, dan zou het niet alleen veel te lang duren, tenslotte zou slechts de allerlaatste kode als gevonden voorwerp worden geregistreerd. Dat is dus niet de bedoeling. Daarom wordt (R*32+K) bij A geteld als zoekadres voor VPEEK. Waar een kode gelezen moet worden, is daarmee geheel afhankelijk gemaakt van de plek, waarop zich de aanwijssprite bevindt. Het kan zich althans in dit geval, ook beperken tot één kode. Immers, die toetsjes zijn samengesteld uit vier kodes. Zo zouden anders voorwaarderegels nodig zijn als: IF C> X AND C<Y THEN . . . Zo kan volstaan worden met het bondige: IF C=Y THEN . . . Dat scheelt weer wat op die toch al talrijke voorwaarderegels. Hier eigenlijk ook onontkoombaar. En aangezien we uitsluitend geïnteresseerd zijn in de kodes zoals met die toetsen verbonden, wordt het omringende gebied voor VPEEK uitgeschakeld in de regels 420 en 430. Wat u ook van het scherm af wilt laten lezen, u zult daarin met deze regels altijd slagen. Hooguit zal dat delen door 8 om R en K te verkrijgen soms iets moeten worden aangepast door bijvoorbeeld bij R 1 extra te tellen of juist af te trekken.
Wanneer u eenmaal zover bent met uw programma, dan is het verstandig om een tijdelijke PRINT-regel in te lassen. Bijvoorbeeld PRINT "C="; C. Zo kunt u dan kontroleren in hoeverre kodes korrekt worden gelezen. Eventueel is er dan nog iets bij te stellen aan het leesadres. Zo'n hulp PRINT-regel is altijd handig. Zeker in een wat ingewikkelder programma, waarin het om een reeks van waarden gaat, die ook allemaal moeten kloppen.

Wat anders alleen via soms vele voorwaarderegels bereikt kan worden, dat is aanzienlijk eenvoudiger, korter én sneller te realiseren middels schermlezing. En u zult het langzamerhand met me eens kunnen zijn, dat ook dit helemaal tot de basis van Basic gerekend dient te worden.

Naderhand, als we samen een programma zullen opzetten, komt ook deze schermlezing weer uitvoerig aan de orde. Maar met wat u er nu van weet, zou u al meteen aan de slag kunnen in een eigen programma.

hoofdstuk 7
Bewegen zonder sprites

Een slechte inval zou het niet zijn, indien u eerst eens een tijdje achter de computer zou plaatsnemen. Om in een "wegwerp"-programma dat VPOKEn en VPEEKen in de praktijk te brengen. Gewoon om voor u zelf vast te stellen, welke resultaten ermee te bereiken zijn. Moeilijk is het niet. Zelfs ik heb het kunnen begrijpen. Anders gaan we maar verder met programma nummertje 6. U mag weer eens typen. U vindt de listing in APPENDIX-VI onder de naam MOVES.
Ik houd niet zo van engelse woorden, als het gebruik ervan niet strikt nodig is. Inplaats van "COLOR3" had ik eigenlijk gewoon "KLEUR3" moeten kiezen. Alleen is "BEWEGING" net even te lang. Vandaar "MOVES". Laten we dat engels beperken tot dergelijk gebruik. Wat is er verkeerd aan ons nederlands? Het maakt een programma echt niet professioneler als bijvoorbeeld PRESS KEY of PLAY AGAIN wordt gebruikt. Mochten we krap zitten in de schermruimte zitten, dan zou bij uitzondering toevlucht kunnen worden gezocht bij het engels, als een woord zich in die taal in minder letters laat uitdrukken. Engelsen of Amerikanen gebruiken evenmin nederlandse woorden. Of het moet al dat woord "apartheid" zijn en daarover willen we het hier niet hebben.

Nadat wij het even opgenomen hebben voor onze moedertaal, hebt u inmiddels MOVES in de RAM-adressen zitten. Het is al eerder opgemerkt. In het algemeen zo niet uitsluitend wordt vooral in de handleiding van de fabrikant de nadruk gelegd op sprites en de grafische mogelijkheden van screen2. Ongetwijfeld allemaal heel handig. Alleen lang niet altijd te gebruiken. Dat grafische al helemaal niet in verreweg het beste scherm screen1. Sprites zijn evenmin altijd gewenst.
Bij een meer uitgebreide figuratie op het scherm zijn het juist ondingen. Sprites zowel als dat grafische dwingen tot beperkingen, die helemaal geen noodzaak zijn, zoals u intussen zelf al hebt kunnen vaststellen. We brengen dat veel liever tot stand middels PRINT na het VPOKEn van karakters. Alles goed en wel, hoor ik u nu mompelen, maar ik wil een figuurtje óók over het scherm laten bewegen. En dus . . . geen sprites! U hebt geRUNd en het MENU afgezocht naar een smakelijke hap. Maar met uw welnemen drukken we eerst toets 1 even in. U kent

dat nu voortbewegende zonnetje wel als kodenummer 15 van set 2. Handig te gebruiken hier. Is er toch een sprite van gemaakt? Sprites komt u niet tegen in dit programma en toch zal er van alles bewegen. Wellicht beweegt ons zonnetje zich wat schokkeriger voort dan een sprite dit zou doen. Zo lekker gaat dit met sprites echter ook niet, als we vanuit die noodzakelijke lussen vaak moeten GOSUBben. Geen sprite en toch beweging.

De oplossing van wat geen raadsel zou mogen zijn, zit in de regels 150 en 160. Binnen de lus wordt het zonnetje gewoon gePRINT. En op de plek waar het tevoren stond wordt een spatie gezet. Eigenlijk beweegt er dus niets. Heel simpel en in voorkomende gevallen toch heel goed bruikbaar. Dat zal nog meer het geval zijn, nadat u toets 2 hebt ingedrukt. Eveneens een schijnbeweging. Twee kodes (184 en 185) zijn veranderd in pijltjes. Het ene is hoog en de andere laag in een 8x8 matrix getekend. Door nu beide kodes afwisselend te PRINTen met een wachtlus ertussen om voldoende effekt te sorteren, lijkt het pijltje op en neer te bewegen. Een spatie-PRINT is hier niet nodig. De tweede PRINT vervangt de eerste. Het is goed te gebruiken, als in een programma speciale aandacht wordt gevraagd voor bijvoorbeeld een mededeling. U zult ze nog wel eens tegenkomen in een gebruiksprogramma. Zo brengt u ook zonder sprites wat leven in de schermbrouwerij.

Toets 3 toont een derde mogelijkheid van beweging zonder sprites. De zogenaamde lichtkrant. Daarvan bestaan verschillende versies en dit is er ook eentje. Met een beetje reklame voor de uitgever. Nu verplaatst zich tekst op het scherm. Maar de MSX-karakters kunnen uiteraard in eigen ontwerpjes worden veranderd. Zo zou bijvoorbeeld een landschapje over het scherm voorbij kunnen trekken. Het nadeel hier is alleen, dat het tot één regel is beperkt. Meer regels kunnen wel, maar dan afwisselend en al te fraai is het niet. Een lichtkrant is echter heel goed te gebruiken in een programma. Als er op een volgestouwd scherm slechts één regel meer vrij is voor mededelingen dan is zo'n lichtkrant een uitkomst. U moet de tekst alleen niet al te rap voorbij laten lopen. Die snelheid is te regelen in regel 400. Het hangt af van de waarde, die u geeft aan hier wachtlus J.
Aan tekst kunnen we tot zo'n 200 tekens kwijt en daarmee zijn duidelijke mededelingen te plegen. In regel 430 is aangegeven, hoe u de boel wat kunt veranderen, zodat er een INPUT op het scherm komt. Gaat u zodanig verbouwen, dan maakt u het zichzelf mogelijk eigen teksten in te tikken en over het scherm te laten flaneren. Het is maar een suggestie.

Zoals u al verwacht had, toont het scherm u na toets 4 nog een manier

van beweging. Dat is het laten scrollen van het scherm. Zoiets kunt u naar boven toe toch al zelf veroorzaken. U zou daarvoor bijvoorbeeld onderstaand programmaatje straks eens kunnen intikken.

```
10  FOR I=0 TO 15
20  PRINT TAB (1);"\  \"
30  NEXT I
40  FOR I=15 TO 0 STEP -1
50  PRINT TAB (1);"/  /"
60  NEXT I
70  GOTO 10
```

De kodes 30 en 29 zijn gebruikt in de PRINT. Na het RUNnen gaat er iets als een autoweg zigzaggen over het scherm. Wellicht iets voor een volgend programma kompleet met autootjes en een benzinestation. Voor dit scrollen hoeft u verder niets te doen. Willen we het van boven naar beneden, van links naar rechts of andersom realiseren, dan valt niet te ontkomen aan een stukje machinekode. Drukt u alvast toets 4 in. Het "ruimteschip" dat u zult zien voortbewegen, zou niet eens uit sprites samen te stellen zijn geweest. Het is alweer door karakterwijziging gedaan. Ditmaal echter niet door PRINT op het scherm gezet. Het is direkt geVPOKEd in de schermadressen van BASE(5). Daar leende het zich goed voor en het komt sneller op het scherm.

Dergelijke routines kunt u aantreffen in boeken over machinekode. De hier gebruikte is overgenomen uit Practical Machine Code Programming van Steve Webb. Het is echter aangepast en in een veel hoger adres gezet. Als u een lang programma hebt, dan kunt u met Basic-regels de adressen overschrijven, waarin de machine-bytes zitten. En dan kunnen er rare zaken plaatsvinden na een RUN. Hoewel dit hier niet nodig was, is het aldus in een hoog adres gezet. Dat kan niet zondermeer. Daarvoor gaat er meer veranderen in zo'n machine-programma. Zonder een assembler gaat het niet. Nu kunt u de routine echter ook veilig gebruiken. Zelfs als uw programma heel wat regels gaat tellen.
In regel 470 is te zien, dat de bytes van de machine-routine slechts eenmaal worden gePOKEd, hoe dikwijls u dit programmadeel ook zou willen bekijken. Wat eenmaal gePOKEd is, dat blijft in die adressen, al zou u met NEW het Basic-programma wissen. Slechts het uitzetten van de computer verjaagt die bytes. Overigens is heel behoedzaam te werk te gaan bij het intypen van DATA-regels met machine-bytes. Bij VPOKE zal een vergissing hooguit het bedoelde figuurtje een wat ander aanzien geven. Bij POKE kan de machine snel blokkeren en bent u alles kwijt, dat al aan programmaregels bijeen was getikt. Daarom wordt aanbevolen om in elk geval een programma met machine-kode

voor alle zekerheid eerst te CSAVEn en pas te RUNnen als die handeling is gepleegd. Mochten we ons dan vertikt hebben, dan is het programma tenminste bewaard gebleven.

Net als bij de voorgaande toepassingen van beweging zonder sprites is ook hier het nadeel aanwezig. Wat zich aan tekst of decoratie op dezelfde lijn bevindt van het bewegende, dat wordt óf "opgegeten", gewist, óf zoals hier meebewegen. U kunt dat vaststellen door regel 725 toe te voegen met: LOCATE 20,6:PRINT "uw naam". Doet u dat ook nog dan zult u de tekst met dat ruimteschip zien meebewegen.
Een bezwaar behoeft dat overigens niet te zijn. Sprites zijn er ongevoelig voor en die kunt u bijvoorbeeld dat schip laten binnen gaan om er lading vandaan te halen. Op soortgelijke wijze is wél een doorlopend landschap onderaan het scherm te realiseren. Zou dit in drie lagen worden ontworpen met WIDTH 32 in screen1, dan zult u per voorstelling aldus 3x32 figuurtjes in een 8x8 matrix dienen te ontwerpen en de verkregen patroonwaarden in de nodige DATA-regels moeten stoppen. Wanneer u dan ook nog een tweede en derde landschap set ontwerpt, dan zal middels READ de tweede vanzelf op de eerste volgen en inderdaad, de derde weer op de tweede. Is de allerlaatste DATA-waarde dan gelezen, dan laat u het van voren af aan beginnen door met RESTORE de DATA-pointer weer op de eerste DATA-regel te zetten. Zo lijkt een eindeloos landschap met bergen, bomen, huisjes of wat u maar zult ontwerpen, voorbij te trekken. Veel werk is het wel. Het mag voor u evenwel niet langer tot de onmogelijkheden behoren. De machine-routine moet er voor gebruikt worden. Daarmee wordt zo'n landschap telkens een plaats opzij geschoven. Zo zal er een plekje vrij komen voor een nieuw stukje. Hiervoor behoeft alleen regel 730 te worden aangepast. Bij: FOR P=20 TO 22 zal uw landschap zich langs de drie onderste schermregels voortbewegen. Maar liever nog wilt u daar een kleurenbalk van zeg maar drie regels dik neerzetten in de kleur van dat landschap. Dan lijkt het allemaal wat dikker en kunnen we ook wat tekst in zo'n balk kwijt. En u weet het nu: regel 23 met dat kleurblokje VPOKEn in BASE(5).
Uiteraard wordt dat FOR P aangepast bij de dán te scrollen regels. Verder zal er niets veranderd worden in deze regel en al helemaal niet in die machine-bytes. Zelfs dat moet u echter helemaal zelf weten. Het is tenslotte uw computer die op hol zal slaan.

Met MSX bent u aldus niet voor één gat te vangen. Al hebt u een ruimteschip van het dubbele formaat voor ogen staan, te maken is het ding en voort te bewegen eveneens. Zonder sprite. Desnoods zelfs zonder machinekode door te VPOKEn in BASE (5) met opschuivende schermlokaties. Er kan heel veel. De beste methoden behoeven niet altijd

benut te worden. Wat geschikt is voor het ene programma, daar zal een andere manier het weer beter doen in een ander programma. Zaak is slechts die methoden te kennen en dit weten in de praktijkervaring nog uit te breiden. Zo zal uw programma niet te snel stuiten op het "onmogelijk" van de computer.

hoofdstuk 8
Sprites en animatie

Al is er met VPOKE en VPEEK een komplete dimensie toe te voegen aan het programmeer-plezier, we willen andersom onze sprites evenmin tot stiefkinderen maken. Het zijn handige dingen, waarmee zo het één en ander te doen is. En de mogelijkheid om sprite-ontwerpen in eigen adressen te stoppen zonder karakters aan te spreken, is een grote plus in MSX. Hier evenwel kan een woord van kritiek op de ontwerpers niet uitblijven. We missen namelijk tenminste twee heel belangrijke zaken bij de MSX-sprites. Hierdoor is het werken ermee in veel gevallen geen onverdeeld genoegen. Eerst al is een sprite uitsluitend over het scherm te bewegen binnen een lus. Waarvan we de waarde toekennen aan X of Y of voor kronkels aan beide. Want al VPOKEn we deze wisselende waarden direkt in de adressen van BASE(9) of fabriceren we een stuk machinekode hiervoor, in alle gevallen is herhaalde aktie noodzakelijk. Het ontbreekt in de PUT SPRITE instruktie eenvoudig aan de mogelijkheid, om daarin óók een plus of min snelheid te noteren. Dán zou de sprite zich onafhankelijk van een lus kunen voortbewegen. En omdat er vanuit die lus dikwijls naar allerhande subroutines moet worden gesprongen, komt dat het soepele voortbewegen van een sprite bepaald niet ten goede. Het is zondermeer een misser, dat zo'n onafhankelijke snelheidsbepaling niet is ingebouwd. Voor beginners is het vaak toch al een raadsel hoe anderen zo'n sprite wél weten te bewegen. Nee, die oplossing van de lus is bepaald onelegant. Hierdoor is er met de MSX-sprites veel minder mogelijk dan anders het geval zou zijn geweest. De tweede misser van de ontwerpers is net zo vervelend. Voor de registratie van een botsing is er ON SPRITE GOSUB voorafgegaan door SPRITE ON. Maar welke van de op het scherm aanwezige sprites is nu het slachtoffer van die botsing? We kunnen niet gebruiken: ON SPRITE (1) GOSUB. Dat zou prompt een foutmelding opleveren. Stel dat we een programma hebben, waarin tussen witte ganzen een zwarte over het scherm vliegt. De witten mogen geschoten worden. Wie de zwarte treft, wordt beboet met zeg maar 10 patronen minder te verschieten. MSX biedt hier geen mogelijkheid om op direkte en eenvoudige wijze die zwarte gans te markeren. We moeten het heel ingewikkeld doen door binnen die lus voortdurend de positie van het zwarte beest te registreren met alle IF . . . THEN regels vandien. Dat is

niet plezierig werken. Eigenlijk zou er nog een derde fasciliteit bij sprites behoren te zijn. Ware het niet dat tóch al met een lus gewerkt moet worden. Dat is de lokalisering van een sprite op het scherm. Door de coördinaten via zo'n instruktie op te vragen, zouden we bijvoorbeeld de sprite in kwestie van richting kunnen laten veranderen. Maar in de lus kan dit ook door: IF I=. Ontbreken doet ook de instruktie, waarmee we de snelheid van bepaalde sprites kunnen wijzigen vanuit het programma. Bovendien biedt ON SPRITE geen marge. Sprites dienen elkaar eerst te raken alvorens het aktief wordt. We kunnen niet bepalen, dat er aktie moet volgen, als de ene sprite bijvoorbeeld nog 8 pickels bij een andere vandaan is. Dán zou er meer spanning in te brengen zijn.

Deze in MSX ontbrekende mogelijkheden voor sprites bestaan wel. In mijn oude computer zaten ze. Erg jammer dat Microsoft ze niet geadopteerd heeft. Dan hadden we veel meer plezier kunnen hebben aan de sprites. Maar goed, misschien komt het nog een keer. Laten we overgaan tot de orde van ons onderwerp hier. We komen weer aan het nodige typwerk voor u. U kunt beginnen aan programma nr. 7 SLALOM. Hierin en in het volgende programma gaat het over sprites. Indien u dat nog niet begrepen mocht hebben. Ditmaal behoeven er geen toetsen te worden ingedrukt. In dit programma zitten twee onderdelen. Ze volgen elkaar vanzelf op. Het is tenslotte af te breken met CTRL+STOP. Het gaat er onder andere om, dat we voor sprites niet per se eerst een figuur behoeven te ontwerpen. Alle karakters, bijna alle, van de MSX-sets kunnen als sprites op het scherm komen. Daarvoor gaan we te werk als in het programma LETTER.

```
A=BASE (9) :B=BASE (7) +8
FOR I=0 TO 7
C=VPEEK (B+I)
VPOKE A+I, C
NEXT I
```

A aldus als start van de sprite-patroon-adressen en B van die van de karakters. Dat bij die B 8 wordt bijgeteld, kan u duidelijk maken, dat we de patroonwaarden van kodenummer 1, het lachebekje, in die sprite adressen gaan VPOKEn. Ze worden eerst weer gelezen en aan C toevertrouwd en dan als C geVPOKEd. In dit programma is hetzelfde gedaan met alle hoofdletters.

U ziet, als het goed is, eerst twee van die lachebekjes over het scherm gaan. Beide hebben dezelfde X en Y waarden gekregen. Toch beweegt de gele zich royaler dan de groene. Dit komt door het gebruik van

STEP in de PUT SPRITE instruktie. De enige niet bijster aantrekkelijke mogelijkheid om een sprite met wat meer snelheid ruimere bewegingen te laten maken. Voor mij had STEP achterwege kunnen blijven. Want daardoor wordt de positie van een sprite in de lus alweer veel moeilijker te bepalen. Als we een tweede sprite sneller willen laten gaan, dan voegen we simpelweg een desgewenste variabelewaarde toe aan X of Y. Vergeet dat STEP liever.

In het tweede gedeelte van het programma komen die hoofdletters als sprites op het toneel. Ze spelen slalom met de lachebekjes in een fatale hoofdrol. Klamme handen zult u er niet van krijgen. Ik heb u alleen even willen laten zien, dat ook karakters tot sprites gepromoveerd kunnen worden. Voor het geval u dat niet wist. Het zou allemaal tot een heel aardig vangspelletje uit te bouwen zijn geweest. Als die twee essentiële mogelijkheden maar niet in MSX hadden ontbroken. Natuurlijk hebben we nog de instruktie SWAP. Dan moet het alleen wél zo uitkomen, dat we een sprite rechtsomkeerd willen laten maken. Er wordt u echter ook nog iets anders getoond, dat allicht te gebruiken is in een programma. Tot sprites gemaakte letters in dubbele omvang zijn bijvoorbeeld ook als titel van uw programma te gebruiken. Al kunnen er dan maar vier op een rij. Vertikaal bepaalt alleen het scherm de grens. Zo is er nog wat leuks mee te doen. Dit overhevelen van karakterpatronen naar spritepatronen is overigens alleen mogelijk bij screen1. Zoals veel bij het VPOKEn en VPEEKen alleen hier te realiseren is. Het mag nog eens de gebruiksvriendelijkheid van dit scherm illustreren. U raakt die sprites het gemakkelijkste weer kwijt door RUN en meteen weer af te breken.

Hoezeer de kritiek inzake MSX-sprites ook gerechtvaardigd is, het betekent bepaald niet, dat we er bij voorkeur geen gebruik van zullen maken. Integendeel. Ondanks die ernstige beperkingen is er toch heel wat mee te doen.
Dan hebben we het over animatie in ruimere zin dan alleen het voortbewegen van een figuurtje over het scherm. Daarvoor lenen sprites zich veel beter dan gePRINTe figuren. Laten we, al is het ten overvloede, eerst eens die PUT SPRITE instruktie bekijken. STEP doet hier niet eens mee.

PUT SPRITE A, (X,Y), kleur, B

A staat voor het vlaknummer en kan variëren van 0 t/m 31. B is het nummer van de sprite. Bij de kleintjes van 0 t/m 255 en bij de grote broers van 0 t/m 63. Die nummering wordt aldus bepaald door de volgorde waarin de spritepatronen in de adressen van BASE (9) worden

gezet. De instruktie SPRITE$ zit intussen al in het rariteitenkabinet, nietwaar? Een vlaknummer moet altijd worden gegeven. Het sprite-nummer kan achterwege blijven. Bij aanwezigheid ervan wordt hiervoor het vlaknummer aangenomen. Beter is het echter om wél het sprite-nummer te noteren. Want zou het om spritenummer 99 gaan, dan kan het vlaknummer niet meer vervangen. Aan het spritenummer zullen we steeds herkennen om welk figuurtje het gaat. Bij het vlaknummer gaat het er niet zozeer om in welke volgorde sprites op het scherm komen. Als u als eerste vlaknummer 0 en als tweede nummer 31 gebruikt, dan houdt dit niet in, dat sprites met tussenliggende nummers vóór nummer 31 op het scherm komen. Zo wordt het soms wel voorgesteld. We bepalen die volgorde zelf via de PUT SPRITE regels ongeacht de te geven vlaknummers. Bij die vlaknummers gaat het uitsluitend om de 32 transparante lagen of stroken, die over het scherm worden gelegd bij het gebruik van sprites. Er kan maar één sprite op zo'n laag. Wel kunnen er 32 achter elkaar, zij het dan niet op één regel. Het vlaknummer bepaalt welke sprite op welke laag gezet zal worden. Zo kan met een gerust hart desgewenst eerst laag 32 gebruikt worden door vlaknummer 31 aan de sprite toe te kennen. Maar geven we een tweede sprite hetzelfde vlaknummer, dan neemt die gewoon de plaats in van de eerste. Omdat wij dus tóch ook het spritenummer hebben genoteerd, zal het ons niet gebeuren, dat de verkeerde sprite met de eer gaat strijken. Nu, daarmee is wat te doen. Daarin zit het hele geheim van de animatie opgesloten. Het vervangen van sprites door aan diverse spritenummers hetzelfde vlaknummer te geven. Dat wil gedemonstreerd zijn om u op de nodige ideetjes te brengen. Onder de naam ANIMAT zit programma nr. 8 er al op te wachten om door u te worden ingetypt.

Ook hier twee onderdelen. Speciaal wanneer in screen1 figuratie op het scherm is gezet middels PRINT, dan kan via een simpele truuk toch gebruik worden gemaakt van ON SPRITE. Hierdoor geeft het de indruk, dat een sprite ook kan botsen met een gePRINTe figuur of zelfs met een verder lege schermplaats. Zou die sprite een voetbal zijn, dan is de muur waartegen het ding te botsen heeft handiger met PRINT op het scherm te zetten.
Zo kunnen we de muur makkelijk uitbouwen tot een kompleet huis met een venster, vanwaaruit een ontstemde vader waarschuwend naar buiten kijkt. Dat gaat niet zo best met sprites. Voetbal-sprite botst tegen muur of venster en via ON SPRITE vindt er een aktie plaats. We kunnen de vader, gekonstrueerd uit sprites, naar buiten laten komen met z'n linker pantoffel zwaaiend. Van alles kan. Toch zijn muur en venster gewoon gePRINT. Er zou eigenlijk geen botsing geregistreerd kunnen worden. Een ander voorbeeld is aan te treffen, als u toets 2 indrukt. Na dit scherm keert het programma vanzelf terug naar het

MENU.

U kent dat wel. Een braaf puffend locomotiefje uit het jaar 0 stoomt vredig voort door een Alpenlandschap. De machinist moet op z'n hoede zijn voor overstekend wild. Zoiets komt voor in zulke streken. Opeens duikt er dan ook een ietwat kloek uitgevallen hert op. Op de rails uiteraard. De machinist moet tijdig en dus op enige afstand van het roekeloze beest stoppen. Zo gebeurt het dan ook. Dat hert botst niet met de loc en daarmee is geen ON SPRITE te aktiveren. Drukt u nu de ESC-toets in, dan ziet u hetzelfde tafereeltje nog eens verschijnen. Duikt de hert op, dan stopt het locomotiefje. Alleen blijken er opeens twee herten te zijn. De tweede veroorzaakt de aktie. Bovenaan is dat evenzeer het geval. Maar daar is dat andere hert niet te zien. Het is transparant geschilderd. Dat is de hele truuk. Op dezelfde manier kan een transparante sprite in dat muurtje worden verstopt. Zo ook is een bepaald schermgebied af te bakenen, als een sprite daar niet zou mogen komen. Gebeurt dit toch, dan volgt de botsing met de niet waarneembare sprite en daarmee de aktie. Subroutines kunnen in beweging worden gezet. Het hoeft niet altijd uit te lopen op een explosie. Op dezelfde manier kunnen we transparante sprites laten meelopen met een zichtbare. Zodanig, dat de zichtbare net iets meer bewegingsvrijheid heeft. Zodat bij een te onvoorzichtige beweging de botsing met een transparantje zal plaatsvinden. Hierdoor is een sprite-figuur bijvoorbeeld door een kronkelende tunnel te leiden. Raakt het de tunnelwand, dan gaat het ding aldus naar de knoppen. Overigens bestaan hiervoor meer methoden, zoals die ook bij een doolhof worden toegepast.

Nu is het gekozen voorbeeld met het locomotiefje niet het meest intelligente. Eerst al bestaat het ding uit twee sprites. Daarom moeten beide delen zich los van elkaar bewegen, wil er niet voortdurend een botsing geregistreerd worden. Het is opnieuw dat manko bij MSX-sprites. Want hoe maken we duidelijk, dat een botsing van de loc-delen genegeerd dient te worden. We kunnen dat niet. Voorts zouden ON SPRITE en het transparante hert hier niet nodig zijn geweest. Immers, de locomotief beweegt binnen een lus.
Als we daarin dan gewoon noteren: IF LUS=X THEN STOPPEN of zoiets, dan had het nog beter gefunktioneerd. Maar ja, wat moest ik dan met dat locomotiefje? Het gaat natuurlijk om het gebruik van transparante sprites.

Het MENU is intussen terug en toets 1 is een drukker te geven. Is het scherm er eenmaal, dan ziet u naast de kaders de cijfers staan voor in

te drukken toetsen. U kunt óók weer toets 2 indrukken en terug gaan naar ons locomotiefje of toets 7 om de voorstelling te beëindigen. Er wordt heel wat afgedrukt in deze programma's, nietwaar? INKEY$ is echter heel handig in het gebruik. Ja, het steekt de tong uit, wil jongleren op een éénwieler, sjokt traag voort, het vliegt, wordt afgeschoten en het belandt in de braadpan. Dat kunnen alleen maar sprites zijn. We gaan het hebben over animatie. U hebt nog even niet gedrukt. In de kaders ziet u de spritesmodellen, zoals voor de animaties worden gebruikt. Daarvan mag alvast één ding meteen duidelijk zijn. Hoe meer modellen, hoe "natuurlijker" de bewegingen. Het kent echter niet louter de grens van het tekentalent. Ook een 16x16 matrix is beperkt en al te veel modellen zijn er nu ook weer niet te maken. Het eenvoudigste is het met een animatie, waarvan de figuur op dezelfde plaats blijft. Zoiets zou ook nog via PRINT te realiseren zijn. Met het eerste voorbeeld is dit het geval. Het beertje zonder lijf. De 6 modellen die zijn gebruikt, hadden hier betrekkelijk ongelimiteerd kunnen worden uitgebreid. Drukt u maar even toets 3 in. Een toontje geeft het einde van de voorstelling aan. Toch heel aardig? Hoe we dit voor elkaar krijgen? Denk aan die vlaknummers. We hebben wél 6 sprites met even zovele spritenummers. Door deze sprites na steeds een wachtlus afwisselend hetzelfde vlaknummer te geven, neemt de ene sprite de plaats in van de andere. Door de identieke vormgeving van de kopjes blijkt deze wisseling niet. De volgende sprite wordt aangezien voor een grimas van het beertje. Dat is met het tweede voorbeeld als toets 4 net zo.

Het is wat ingewikkelder. Er moeten twee sprites boven elkaar voortbewegen en beide sprites moeten worden afgelost door andere modellen. Nu zou het eigenlijk meer snelheid moeten zijn gegeven om dat wiel beter tot z'n recht te laten komen. In dat geval zou het minder geloofwaardig zijn geworden, dat het mannetje zich met zulke woeste bewegingen wist te handhaven. Als het even kan, moeten de verhoudingen een beetje kloppen bij animatie. Overigens mag een en ander u ervan overtuigen, dat met sprites niet louter ufo's en ander ruimtetuig te fabriceren zijn. U ziet het ook aan de olifant, die z'n entree maakt na toets 5. Hiervoor zijn 8 sprites gebruikt. Ook met een tweede kleur met de uitsparing daarvoor in een andere sprite. Meer kleuren kan natuurlijk wel, maar het dient wél goed tot zijn recht te komen.
Het oog van de olifant bijvoorbeeld groen te maken zou jammer van de moeite zijn geweest. Je ziet er amper iets van terug. Met dat wat bescheiden uitgevallen rugdekje gaat het nog. Het benenwerk van jumbo is niet ideaal. Eigenlijk zouden daarvoor meer modellen nodig zijn geweest. Hier zijn twee sprites boven elkaar en twee naast elkaar gezet. Het nadeel van dit achter elkaar plaatsen van speciaal 16x16

sprites is, dat in dit geval de olifant hevig geamputeerd wordt bij het bereiken van de linker schermrand. Want eerst springt het voorstuk weg en even later de rest. Om u dit afgrijselijke gebeuren te besparen, is helemaal links dat grijze blok neergezet. U kunt er zich van overtuigen door in regel 170 voor de laatste LINE-instruktie even REM te zetten en nog eens te RUNnen. Dat grijze blok is weg als jumbo die hoek nadert . . . u zult moeten toegeven, dat het geen gezicht is. Aldus maar snel dat REM weer weghalen.

De vierde animatie is de meest simpele. Het bestaat uit feitelijk slechts twee modellen en doet het toch heel aardig. Toets 6 veroorzaakt hier de aktie. Het dient een gans voor te stellen. Weliswaar ondersteunt geluid de animatie heel goed, maar ik heb met SOUND geen betere gakgak weten te produceren. De SOUND-instruktie van MSX is zo veelzijdig en uitgebreid, dat het niet zo simpel is om te laten klinken, wat je graag wilt horen. Het zal een kwestie van meer ervaring en vooral ook van meer muzikaal gehoor zijn. Nu, de animatie voert hier de boventoon. Laten we het voor bijvoorbeeld de gans nog eens bijeen zetten.

```
PUT SPRITE 1,(X,Y),15,0
FOR D=1 TO 50:NEXT D
PUT SPRITE 1,(X,Y),15,1
```

Y heeft een vaste waarde en X wordt bepaald in de lus. Sprite 0 en 1 wisselen elkaar af door hetzelfde vlaknummer. Omdat er 32 vlaknummers zijn, kunnen we daar normaal wel even mee vooruit. In het programma staan echter ook de modellen op het scherm. Dat zou niet kunnen zonder ze een eigen vlaknummer te hebben toegekend. Zo houden we hier voor de animatie niet al te veel vlaknummers over. Om precies te zijn 9 en net niet genoeg. Wanneer die rode gans bezig gaat omlaag te vallen, dan ziet u het berekopje helemaal linksboven verdwijnen. Ik heb zijn vlaknummer moeten lenen ten koste van zijn verschijning. Zodra evenwel die gans in de braadpan zit en met de 209 is afgeschreven, komt het berekopje weer terug. Dus is het bij animatie vooral een zaak om die vlaknummers in de gaten te houden. Daar draait het allemaal om. We mogen overigens een lusvariabele gerust een hoge waarde meegeven. Voor de schermbreedte hoeft dat niet per se 0 TO 255 te zijn. In het betreffende register krijgt de byte vanzelf weer nulwaarde zodra de hoogste waarde is bereikt. Vandaar de indruk dat een sprite als het ware achter het scherm langs gaat. Maar wat in een lus kan, dat is niet toegestaan als aan X en Y absolute waarden worden gegeven. Dan dienen we ons aan de voorschriften te houden. Het zal niet zo vaak voorkomen in een programma, dat er zo-

veel sprites bijeen zijn en daarvan ook nog een stel moeten animeren. In het programma is de grens zo ongeveer bereikt. Dat zulke animerende sprites evenzeer in kombinatie met gePRINTe figuren zijn te gebruiken, dat zal inmiddels al een overbodige opmerking zijn geworden. Alleen wat dient te bewegen kan uit sprites bestaan. De voorbeelden mogen aanmoedigen het zelf ook eens in praktijk te brengen. Het hoeft immers niet meteen in een programma. Als we maar eenmaal weten, hoe iets tot stand kan komen, dan beginnen we al met heel wat opgewektere gevoelens aan een groot programma.

Wat zoal is aangesneden aangaande VPOKE, VPEEK, het printen van de karakters, beweging en sprites, zal samen met de goede toepassing van de overige Basic-instrukties hoe dan ook tot een goed programma moeten kunnen leiden. Of tot een geheel ander soort programma dan u tot nu toe gewend was te maken. Ons resten nog twee programma's. Dat zijn niet zozeer demonstratieprogramma's. De laatste wordt dus gebruikt, om stap voor stap en soms met grote stappen samen een programma op te zetten. Dat zal vooral goed zijn, als u er moeite mee hebt om een programma op te bouwen. Tegelijk is het als een analyseren van een programma. En dat zullen we regelmatig doen met andermans programma's. Daarvan kan veel worden opgestoken. We blijven evenwel eerst nog even in de buurt van de sprites. Of veel beter uitgedrukt: bij dat zelf ontwerpen van figuren, die al dan niet op het scherm dienen te bewegen. Hoe doe je dat het beste en heb je er nu echt zoveel tekentalent voor nodig? Dit laatste kan alvast ontkennend worden beantwoord.

hoofdstuk 9
Figuren ontwerpen

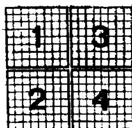
Bij VPOKE gaat het meestal om zowel kleur als vormverandering van één of meer karakters. We willen een zelf verzonnen figuurtje hebben. Op zichzelf staand of als onderdeel van een uitgebreide schermdecoratie. Als sprite en veel vaker om te PRINTen daar waar we het willen hebben. U treft in APPENDIX-V zowel een ontwerpvel voor screen1 als een matrix-vel aan. Daar van zijn kopieën te maken. Het matrix-vel zult u in de praktijk vooral gebruiken om ruwweg aan te geven hoe een figuur er zou moeten uitzien. Voor het eigenlijke ontwerpen mag ik u een programma aanbieden, dat in eigenlijk alle behoeften bij dit ontwerpen voorziet. Het is dan ook wat anders dan zo'n sprite-editor, dat u wellicht kent. Alle in de programma's gebruikte figuren zijn met dit ontwerp-programma tot stand gekomen.
Eerst iets over dat intekenen van een matrix. Dat het niet moeilijk maar wel lastig kan zijn, komt door het effekt dat een ontwerp op het scherm zal geven. Echter ook een Rembrandt lijkt van een afstand beter, dan wanneer we er met de neus bovenop staan. Iets dergelijks geldt voor onze elektronische Rembrandtjes. Wat eenmaal getekend in een matrix nergens naar schijnt te lijken, dat kan op het scherm juist prima te voorschijn komen. Daarin zit meteen ook de last. Bevalt het ons niet, wat op het scherm komt, dan is de betreffende DATA-regel te korrigeren. En dat kan zo een tijdje doorgaan, als we tot de perfektionisten behoren. Bij een enkel figuurtje is het nog wel te doen. Gaat het om een samenstel van figuurtjes, dan wordt het allemaal nog lastiger. Sommige van die editors geven wél dat ene ontwerp weer, zodat het op het scherm te beoordelen is. Maar wat als onze figuur zal bestaan uit 20 matrices van 16x16? Dan willen we het ook graag als één geheel kunnen beoordelen in een weergave op het scherm. En waar nodig van ontwerp naar ontwerp kunnen springen om korrekties uit te voeren. Als we dan tenslotte de DATA-regels, sekuur, intikken, dan krijgen we precies, wat we ons wensten. Maar zo'n editor bestaat er niet. Of toch wel?

Programma nr. 9 vraagt weer eens het één en ander van uw doorzettingsvermogen en zorgvuldigheid bij het intypen van een listing. Het zal nu zeker alle moeite waard zijn. De listing is de naam PRENT meegegeven.

Want om prentjes gaat het hier helemaal. Voortaan ook door u zonder al te veel moeite te ontwerpen. Nodig is een beetje fantasie en zorgvuldigheid en het vermogen de cursortoetsen en de spatiebalk te kunnen indrukken. Het wordt u zondermeer toevertrouwd. Hiermee zijn niet alleen relatief snel figuren te ontwerpen in formaat 8x8 of 16x16, uit meerdere formaten zijn grote figuren vrijwel probleemloos samen te stellen. Het enige "handwerk" hierbij is het noteren van de DATA-regel(s). En als u een printer hebt, zou zelfs dat te omzeilen zijn. Het is overigens niet in het programma opgenomen. Al te zinvol is het niet. Tenslotte zult u hoe dan ook toch de DATA-regel zelf in uw programma moeten typen. Dan wel met de zekerheid, dat u deze niet meer behoeft te korrigeren. Want dat is evenzeer mogelijk met het programma.

Eigenlijk is het niet zo nodig toelichting te geven bij het gebruik van het programma. Het is voldoende duidelijk op het scherm aangegeven. De gelegenheid is er hier echter om wat uitvoeriger te zijn. Het programma is vrij simpel van opzet. Het zou ook aanzienlijk beknopter hebben kunnen zijn dan soortgelijke programma's. Er is evenwel voor een prettig werkend scherm en een uitbuiting van de mogelijkheden gekozen. Slechts terwille van die gebruiksvriendelijkheid is het langer geworden, dan strikt noodzakelijk zou zijn geweest. Door het gebruik van vrij veel variabelen is het wat te ingewikkeld geworden voor een uitvoerige bespreking.
Het gaat helemaal uit van de schermlezing, waarmee u al kennis maakte bij CALCUL. Hier is de aanwijssprite een soort spinnetje. Waar het ding staat zal via VPEEK worden gelezen om welke kode het gaat. Als u daarvoor althans de spatiebalk hebt ingedrukt. Gaat het dan om een kodenummer, waaraan een routine is verbonden, dan zal dit de start van allerlei akties worden. Waar zo'n kode voorkomt, is dus van geen enkel belang. Gelezen wordt alleen de plaats, waar het spinnetje zich bevindt. Zo zal dit programma u er eveneens van mogen overtuigen, hoeveel er mogelijk is met deze schermlezing in BASE (5).

U zult na de RUN een uitnodigend schermbeeld voor u zien. Goeddeels ingenomen door een 16x16 matrix. Het is in vier sekties verdeeld; a, b, c en d. Het is ook de volgorde, waarin de patronen bij sprites in de adressen van BASE (9) dienen te worden geVPOKEd.



Ofschoon deze volgorde bij het PRINTen niet nodig is, wordt dit hier wel van u verwacht. Het maakt niet veel uit. Het komt immers neer op:

```
LOCATE X,Y:LOCATE X+1,Y
LOCATE X,Y+1:LOCATE X+1,Y+1
```

En bij een 16x16 matrix gebruiken we uiteraard:

```
LOCATE X,Y: PRINT "ac"
LOCATE X,Y+1:PRINT "bd"
```

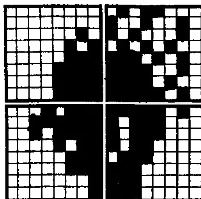
het handige bij zulk PRINTen is, dat na een eerste RUNning van het programma de gebruikte MSX-kodes voor ac en bd veranderd zijn in uw ontwerp. Hierdoor is meteen te kontroleren of u wel de juiste kodes hebt ingetypt. Nemen we bijvoorbeeld even aan, dat de karakters 128 t/m 131 zijn gebruikt voor dat ontwerp, dan wordt die PRINT-regel aldus:

```
LOCATE X,Y:PRINT "çé"
LOCATE X,Y+1:PRINT "üâ"
```

Zolang dit er staat, kan het onzeker blijven of deze 4 karakters ook werkelijk samen de vorm van ons ontwerp hebben aangenomen. Na de RUN hoeft dat niet meer. En ook bij het afbreken van een programma, zult u bij het LISTen het ontwerp weer tegenkomen in plaats van die 4 karakters. Na een paar maal werken met VPOKE is die onzekerheid er niet meer. Dan zult u er gewoon blindelings op vertrouwen, dat alles op het scherm komt, wat u daarvoor hebt ontworpen. Mits u natuurlijk middels de juiste toetsen ook de juiste karakters in de PRINT-regels hebt gezet. We moeten echter nog even zo'n ontwerp maken.
Voor een 8x8 formaat wordt alleen sektie a van de scherm-matrix gebruikt. Is eenmaal gekozen voor 8x8 of 16x16, dan zitten we daar min of meer aan vast. Het is niet mogelijk om zondermeer van het ene op het andere formaat over te stappen. De voornaamste oorzaak hiervan is, dat het ontwerp als een sprite wordt weergegeven. Verbazen hoeft u dit aldus niet. In een programma kunnen evenmin beide formaten naast elkaar worden gebruikt. De zaak is overigens beveiligd tegen eventuele willekeur. Het zal in de praktijk weinig voorkomen, dat beide formaten naast elkaar worden gebruikt. Bij het PRINTen van figuren kan dit natuurlijk wel. Wat dan te doen is, is bijvoorbeeld eerst de 8x8 formaten en daarna de 16x16 formaten te ontwerpen. Zijn de eerste gereed en daarvan de DATA-regels genoteerd, dan zet u het spinnetje op het blokje met RET. Even spatiebalken en de zaak wordt

geveegd. U kunt vervolgen met het andere formaat.

Zoals de "rebus" rechtsboven duidelijk maakt, zijn met de cursortoetsen alleen horizontaal en vertikaal het spinnetje vlot te verplaatsen. Diagonaal is hier weinig zinvol en al te fijn werkt dat evenmin met deze toetsen. U ziet die aanwijssprite al wachten linksboven in de matrix. De bedoeling is ook hier, dat u hokjes zwart zult maken, waardoor het patroon van uw ontwerp zal ontstaan. De spatiebalk is ons penseel. Even drukken en zwart is het hokje onder het spinnetje. Blijkt dit naderhand ongewenst te zijn, dan neemt u er weer op plaats. Andermaal spatiebalken en leeg is dat hokje weer. Heel simpel, effektief en vooral snel. We hebben het allemaal duidelijk voor ons op het scherm. Zo is door verplaatsing van het spinnetje en de spatiebalk het gewenste patroon in te tekenen. Laten we gemakshalve uitgaan van 16x16 formaat. U pakt het natuurlijk meteen groots aan! Wanneer nu het zo ingevulde patroon lijkt op de figuur, die u wilt ontwerpen, dan is het prettig om het ding in ware grootte op het scherm te zien verschijnen. Pas dan is goed te beoordelen of het ontwerp werkelijk volgens wens is. Daartoe bewegen we de spin naar het blokje met 16-16 en weer spatiebalken. Nu gaat er van alles gebeuren. Blauwe pijlen wijzen de plaats aan, waar het ontwerp zal worden weergegeven en de patroonwaarden komen de een na de ander als hex-kodes in de juiste volgorde op het scherm. Het ontwerp komt tevoorschijn. In cyaan-blauw. Maar u had ook uit één van de 6 beschikbare kleuren kunnen kiezen. Wanneer u eerst het spinnetje op het rode, witte, groene, blauwe, gele of het magenta-blokje had gezet. Helemaal rechtsboven in dat kleurbalkje. En staat u eenmaal redelijk sekuur op zo'n blokje, dan is andermaal een tikje op de spatiebalk afdoende om die kleur te laten registreren in het programma. Deze kleurkeuze maakt het mogelijk, om het ontwerp in ongeveer de kleur weer te geven, welke u er in uw programma aan denkt te zullen geven.
Maar dan uiteraard wél eerst kleurbekennen en dan op het 16x16 ikoontje het ontwerp laten weergeven. Er kan zo dikwijls van kleur worden gewisseld als het u in de zin komt. Op het zwarte vlak is meer dan één ontwerp weer te geven. Bij de 16x16 formaten kunnen dat er 20 zijn en bij de 8x8 zelfs 28. U kunt er dus even mee vooruit.



Stel dat u dit als indianenkop bedoelde ontwerp zou hebben gemaakt. En het bevalt u nog niet helemaal. Dat zou in dit geval ook logisch zijn, want een gekompliceerd patroon is zelden in eenmaal goed te tekenen. Maar geen nood. We zetten de spin op het ikoontje met de naar links wijzende pijl en drukken op de spatiebalk. Die blauwe pijlen bewegen wel even, maar ze kunnen niet verder naar links. Het zwarte vlak telt voor deze 16x16 formaten dus 20 lokaties. Het nummer daarvan wordt bijgehouden in het kadertje helemaal rechtsonder. Maar zowel op dit nummer als op die blauwe pijlen komen we nog terug. Boven dat pijl-ikoontje is cijfer 1 verschenen; u bent 1 plaats terug gegaan. Als het gekorrigeerde ontwerp straks wordt weergegeven, zal dit cijfer weer verdwijnen. Deze handeling geldt overigens alleen voor de allereerste lokatie. Zodra meer ontwerpen zijn weergegeven, komt er nog een handeling bij in het geval een voorgaand ontwerp gekorrigeerd moet worden.

U bent terug gegaan naar de matrix, hebt wellicht zwarte hokjes gewist en anderen zwart gemaakt, hier mag dat, en denkt bij u zelf: zo zou het kunnen. En nu maar eens in rood weergeven. U weet al, wat te doen. Ook om het zo gekorrigeerde ontwerp te kunnen weergeven. Het minder geslaagde exemplaar verdwijnt en het gekorrigeerde treedt tevoorschijn. In rood. Mocht u dit boeien, dan kunt u het ontwerp zo nodig 20 maal weergeven door even zoveel maal met het spinnetje op dat 16x16 blokje te spatiebalken. Het duurt altijd wel even alvorens dan ook de vier DATA-regels op het scherm staan. Die 20 zijn het maximum. Bent u aan de vijfde rij begonnen, dan maakt ikoontje RET plaats voor een herinnering hieraan. Eigenwijs als we zijn, proberen we tóch nummer 21 weer te laten geven. Tja, dat hadden we nu niet moeten doen. Sprites weg, matrix geveegd, pijlen weg, kortom een schone lei om opnieuw te beginnen. Zo vergaat het u ook als u meer dan 28 8x8 formaten wilt laten weergeven.
U kunt het zometeen altijd nog uitproberen. Het gekorrigeerde ontwerp staat in oorlogzuchtig rood afgebeeld en u bent voldaan. Aldus de DATA-regels op een kladje genoteerd en eventueel aan het tweede ontwerp begonnen. Wachten we er nog eventjes mee. Want al hebt u de goede DATA-regels nu ter beschikking, er dient nu een karakterset te worden gekozen, waarvan u in dit formaat 4 karakters wilt omturnen in het ontwerp. We nemen het nog even door. We kiezen set 17 met de eerste 4 karakters 128 t/m 131. Om het adres te verkrijgen, waarin aan de vorming van karakter 128 wordt begonnen, vermenigvuldigen we aldus die 128 met 8. Een karakter bestaat nu eenmaal uit 8 adressen. De uitkomst behoort 1024 te zijn. Dat wilden we even weten. Het gaat er dus zo uitzien in ons programma:

```
A=BASE (7) + 1024
FOR I=0 TO 31
READ C$
VPOKE A+1, VAL ("&H"+C$)
NEXT I
DATA met de waarden van uw ontwerp
```

En u weet het al; de I-lus moet met 0 beginnen, omdat bij de eerste doorloop niets bij A mag worden geteld. Met 1024 hebben we immers al het startadres kunnen bepalen. Omdat we 4 kodes nodig hebben voor het 16x16 formaat en deze kodes eveneens in 8 adressen zijn opgebouwd, moet via READ de DATA-regel dus 4x8 is 32 keer worden gelezen. Ook in het geval u meer, zelfs veel meer DATA-regels bijeen hebt genoteerd, beperkt u dan in elk geval bij 16x16 één DATA-regel tot die 32 waarden. Wel kunnen er ook 64 in staan, maar daarvoor kan amper een reden bestaan. En door deze beperking komt elke DATA-regel te staan voor één 16x16 ontwerp. We kunnen er zo nodig nog een tijdelijke REM-regel boven zetten, waarin we bijvoorbeeld het ontwerp omschrijven of er een volgnummer aan toekennen. Tegelijk zal genoteerd worden op dat kladje, welke DATA-regel bij welk ontwerp hoort en welke karakterkodes hiervoor zijn gebruikt. Is het eenmaal helemaal goed, dan is zo'n REM-regel snel genoeg weer te verwijderen en dat kladje weggeworpen. We kunnen altijd een typefoutje begaan bij het intikken van vooral veel DATA-regels. Dan is het wel zo prettig om de regel in kwestie snel terug te kunnen vinden. Van deze werkwijze kunt u veel gemak hebben. In bovenstaande programmaregels is uitgegaan van het PRINTen van het ontwerp. Moet het een sprite worden, dan maakt u van BASE (7) alleen BASE (9) en daar hoeft dan niets bijgeteld te worden. U noteert alleen even, dat het om spritenummer 0 gaat. En behalve via PRINT zijn de patroonwaarden eveneens direkt te VPOKEn in de schermlokaties van BASE (5). U hebt dat kunnen nagaan bij dat ruimteschip in het programma MOVES. Nodig is hiervoor alleen, dat we de juiste schermlokaties vaststellen.

U zou het in elk geval eens moeten uitproberen. Om ook met deze methode vertrouwd te raken. Nu is het PRINTen wat duidelijker en hoe dit gaat, dat is in het begin al uit de doeken gedaan. Nog zonder één en ander in een programma te verwerken, lijkt het een goede zaak, wanneer u met de verkregen DATA-regels in een proefprogramma wat zult gaan oefenen met dit VPOKEn. Het moet even vertrouwd worden als de overige Basic-instrukties. Willen we het ontwerp in rood met een zwarte achtergrond op het scherm zetten, dan dienen we natuurlijk vóór de PRINT set 17 eerst deze kleurkombinatie te geven. U kent het al:

```
A=BASE (6)
FOR C=128 TO 135
VPOKE A+(C/8), &H61
NEXT C
```

Altijd de hele set verkleuren. Al gebruiken we er maar vier karakterkodes van. Dat kan niet anders. En daar moet rekening mee worden gehouden, als er meerdere 16x16 figuren nodig zijn in weer andere kleuren. Voor elke kleur is een aparte set nodig. Als het even kan, doen we het niet al te kleurrijk, zodat wél alle 8 karakters van een set te gebruiken zijn. Dat maakt het VPOKEn van de patroonwaarden van die ontwerpen ook veel eenvoudiger. Want al willen we de karakters van 10 sets veranderen in wat we zelf hebben ontworpen aan figuren, dan kan het allemaal in één VPOKE-lus worden ondergebracht. Wat na TO komt, dat wordt bepaald door het aantal te VPOKEn karakters maal 8. Bij 80 stuks aldus: FOR I=0 TO 639 (als 640-1). Alle DATA-regels worden dan achter elkaar gelezen. Evengoed kan dan een hele set een andere kleur worden gegeven. Gebruiken we door veel kleuren slechts 4 karakters van een set, dan is voor elke set wél een aparte VPOKE-lus nodig. Alsof je een open deur intrapt, zult u wellicht denken. Daarop moet het voor u ook gaan lijken. Alleen bij gebruik van sprites hoeft er uiteraard geen rekening mee te worden gehouden. De kleur wordt immers bepaald in de PUT SPRITE instruktie. Te kombineren is het evenzeer. In het geval u een figuur zowel wilt printen én als sprite wenst. Het ziet er alleen wat uitgebreider uit:

```
A=BASE (7) +1024:B=BASE (9)
FOR I=0 TO 31:READ C$
VPOKE A+I, VAL ("&H" +C$)
VPOKE B+I, VAL ("&H" +C$)
NEXT I
```

Het kan in één moeite door. Als alleen de sprite nodig is, dan spaart u natuurlijk de karakters van set 17 en volstaat me B=BASE (9) etc. . . Voor het definiëren van een sprite behoeven we gelukkig geen karakters aan te spreken. Dat VAL en "&H"+ is nodig, omdat alleen decimale waarden worden geslikt. Alleen als we in plaats van hexadecimale de decimale waarden in een DATA-regel zetten, dan kan hiervoor gewoon C worden gebruikt. Maar dat doen we niet. We zijn juist al wat vertrouwder geworden met de hex-kode en hebben er de voordelen van weten te waarderen.

Even duidelijk mag geworden zijn, dat weliswaar niets behoeft te worden bijgeteld bij BASE(9) voor sprites, maar dat we dan wél alle be-

nodigde spritepatronen achter alkaar gaan VPOKEn. Zouden we 20 van die dingen nodig hebben, dan zijn er helemaal geen 20 regels nodig met die SPRITE$ instruktie. Aangezien 20x32 nog altijd 640 is, tikken we gewoon 639 achter TO en de benodigde DATA-regels worden alweer achter elkaar in de adressen van BASE (9) gezet. Ook hier het kladblok bij de hand houden. Bij de genoteerde DATA-regels in de juiste volgorde even de spritenummers vanaf 0 noteren. Anders weten we naderhand niet meer wie spritenummer 17 ook al weer was. Desnoods maken we een tekeningetje van het ontwerp bij zo'n nummer. Zo blijven we geïnformeerd. Wat zorgvuldigheid kan veel tijd en ergenis besparen.

Gaan we evenwel even RETURN naar ons ontwerp-programma PRENT. Het eerste ontwerp is gereed, de DATA-regels netjes genoteerd en er kan aan het volgende worden begonnen. Wanneer nu dit tweede ontwerp niet te gek veel afwijkt van het eerste, dan kunnen we in de matrix nog het beste de ongewenste zwarte hokjes wissen via de spatiebalk. Wordt het iets geheel anders, dan is beter de hele matrix te wissen. Daarvoor is er het WIS-ikoontje. Spinnetje er op, spatiebalken en schoon wordt de lei. Verder verandert er niets. Het eerste ontwerp blijft, waar het werd neergezet. U gaat weer aan de slag en weldra is nummer twee gereed. Het gaat vlot. En omdat u nog even moet wennen, zet u het spinnetje voor de weergave van dit tweede ontwerp al dan niet per ongeluk op het 8x8 ikoontje. Foei! Het intussen bekende pijltje duikt op en wijst nerveus piepend het 16x16 ikoontje aan, waarop u had moeten gaan staan. Echter evengoed wordt het ontwerp in het eenmaal gekozen 16x16 formaat weergegeven. U kunt daarbij de blauwe pijl zien verspringen en onderaan in dat kadertje cijfer 2 zien verschijnen. Het is er niet voor de aardigheid. Nodig is het namelijk niet, om dat tweede ontwerp pal naast de eerste te zetten. U had de spin ook eerst op het ikoontje met de naar rechts wijzende pijl kunnen zetten. Het tikje op de spatiebalk doet cijfer 1 eronder verschijnen en de blauwe pijl verspringt braaf van plaats, waarbij het lokatienummer zich al even braaf aanpast. Het is echter niet helemaal volgens wens.
Zo komen we toch nog naast het eerste ontwerp te staan. Dus nog een tikje en cijfer 2 zal verschijnen. Nu zal de blauwe pijl de gewenste plek aanwijzen. Hieruit zal gekonkludeerd worden, dat het niet nodig is om dit pijl-ikoontje te gebruiken, als het volgende ontwerp gewoon direkt naast het vorige mag komen. Deze pijl-ikoontjes "vooruit" en "terug" gebruiken we alleen, als er meer dan één lokatie opgeschoven moet worden. Deze mogelijkheid zult u gaan waarderen zodra uit meerdere 16x16 of 8x8 formaten één groter figuur moet worden samengesteld. Bij gebruik van PRINT zal dit vaak aan de orde zijn. Het probleem

hierbij is immers, dat de afzonderlijke figuren goed op elkaar moeten aansluiten. Kon er slechts één ontwerp worden weergegeven of een tweede slechts daar direkt naast, dan moet het toch nog op een matrix vel worden nagetekend. Anders weten we niet meer waar we wat op elkaar moeten aansluiten. Met dit programma hoeft dat dus niet. Alleen voor een vlotte werkwijze kunt u voordien zo'n samengestelde figuur ruwweg aangeven op een matrixvel. Dan is alvast te beoordelen met welke lokatie moet worden begonnen en welke lokaties wel of niet zullen worden gebruikt. Echt nodig is evenwel ook dit niet.

U kunt voor zo'n samengestelde figuur meteen voor het eerste ontwerp al bijvoorbeeld twee lokaties overslaan. De blauwe pijl gaat dan de gewenste plaats aanwijzen. In het extreme geval dat u zeg maar alleen op de 3e en de 17e lokatie een ontwerp wilt weergeven, een merkwaardig figuur wordt dat, dan werkt u vanaf lokatie 3 net zo lang met het "vooruit" ikoontje, totdat de blauwe pijlen én het lokatienummer plaats nummer 17 markeren. Er is een wachtlus opgenomen bij dit verplaatsen. Anders zou het ongewenst snel kunnen gaan. Mocht u zich bedenken en toch ook nog een ontwerp op lokatie 9 willen weergeven, dan keert u via het "terug" ikoontje naar die lokatie terug. Op die manier kunt u op elk der 20 beschikbare lokaties een ontwerp weergeven. Al zou u daarvoor van lokatie 20 terug moeten keren naar lokatie 1. Blauwe pijlen en lokatienummer helpen u daarbij een handje. U hoeft alleen telkens de DATA-regels maar te noteren van het geslaagde ontwerp alvorens naar een andere lokatie te gaan.

Gaat het evenwel om het ontwerp dat juist is weergegeven op een lokatie hoger dan 1, dat nog korrektie nodig heeft, dan moet eerst 1 plaats terug en dan weer 1 vooruit worden gegeven. Valt dus wel mee die extra handeling. Het went allemaal snel en u zult ervaren, dat het erg prettig werken is met dit programma. Want het is helemaal afgestemd op de praktijk van vooral het PRINTen van figuren. Zo behoeft dit ontwerpen voortaan geen probleem of tijdrovende bezigheid meer te zijn. Daar gaat het om.
Want wat niet in eenmaal slaagt, dat kan net zo vaak gekorrigeerd worden tot het u wel bevalt. Dat ene resterende ikoontje met END spreekt voor zichzelf. Eenmaal daarop gespatiebalkt en alles verdwijnt als sneeuw voor de zon. Het blauwe standaardscherm kompleet met de funktietoets-teksten keert terug. U hoeft de cursor alleen nog via de HOME-toets op de gebruikelijke plaats te zetten. En wilt u het programma helemaal kwijt ook nog even NEW intikken. En dan nog dit. We hadden het al over dat van een afstand bekijken van prentjes. Mocht u een en ander ook nog tot stand willen brengen vanuit de luie stoel, dan is er niets anders nodig, dan in de regels 130 en 190 achter

STICK en STRIG de (0) te vervangen door (1) of (2). Dan doet u het allemaal met joystick en vuurknop. Het gemak dient tenslotte de mens. Nu, allicht wilt u er nu een tijdje mee aan de slag om de zaken uit te proberen. Zou het niet dat heel nuttige doel dienen van de DATA-regels, dan zou het ook nog een aardig spel kunnen zijn. Op dat zwarte vlak zijn met gebruik van de kleurmogelijkheden aardige dingen weer te geven. U komt er wel achter.

hoofdstuk 10
Programmeren

Laten we ter afronding de besproken zaken eens in de praktijk brengen. Wat gezegd is over de zekere moed, nodig om aan een programma te beginnen dat zal hier aktueel worden. Daarom kunnen we net doen, alsof we samen een groot programma zullen opzetten. Een troost zal het daarbij mogen zijn, dat ik net als u een amateur ben met wellicht hooguit iets meer ervaring in de omgang met de home-computer. Bijvoorbaat betekent dit ook, dat mijn werkmethode niet per se de uwe hoeft te zijn en al evenmin voor de meest efficiënte mag worden versleten. Amateurprogrammeurs lijken hier te kunnen worden onderscheiden in de meer verstandelijke en de meer intuïtieve figuren. De rationelen doen aan stroomschema's en werken alles zodanig op papier uit, dat de boel na het intypen meteen naar wens funktioneert. Wat trouwens lang niet altijd het geval behoeft te zijn. En bovendien een grondige kennis van zaken en veel ervaring als voorwaarden stelt. Ik doe het zelf liever meer intuïtief, hoe betrekkelijk dat ook mag zijn binnen de rationele grenzen van Basic. Is het programma-idee eenmaal ontstaan, dan meteen achter de computer en via AUTO het eerste regelnummer op het scherm zetten. CLS en screen1 is dan altijd een geslaagde start, waaraan we KEY OFF en COLOR nog kunnen toevoegen. Daar laat ik het doorgaans bij. In de volgende regel komt dan alleen GOSUB te staan met een hoog regelnummer; zeg maar GOSUB 1000. Dat wordt dan een REM-regel met bijvoorbeeld "schermbeeld" erachter. Want daar beginnen we mee. Als er tenminste een schermbeeld nodig is, hetgeen meestal het geval zal zijn. Zo'n kaal scherm is ook niets gedaan, hoe zakelijk het programma ook zal worden.
Nu daarmee is het programma voorlopig af. Nog even RETURN in regel 1010 en GOTO 30 in, inderdaad, regel 30. Alleen is er zo weinig lol aan te beleven natuurlijk. In werkelijkheid zullen we alle informatie, nodig voor dat schermbeeld, al kant en klaar hebben liggen, eventueel kompleet met de DATA-regels zoals verkregen dank zij PRENT. Daarom tikken we na die REM-regel 1000 meteen maar één en ander in onze schermbeeld-routine. Als het eerste schaap maar over de dam is. Die routine maken we dus eerst zo veel mogelijk af met allles wat er aan kleurvlakken, figuren en/of tekst op het scherm moet komen. Tenminste, als we niet iets heel moeilijks willen in ons programma.

Dán is het wel zo wijs, om dat moeilijke eerst even uit te proberen in ons programma. Kan het wel en welke truukjes moet ik er voor uithalen. Slaagt het niet omdat wij tekort schieten óf omdat MSX-Basic er simpelweg geen mogelijkheden voor biedt, dan schieten we ook met het fraaiste schermbeeld weinig op. Ik bedoel maar. Het moet je eerst een keer zijn overkomen, alvorens je daar rekening mee leert te houden.

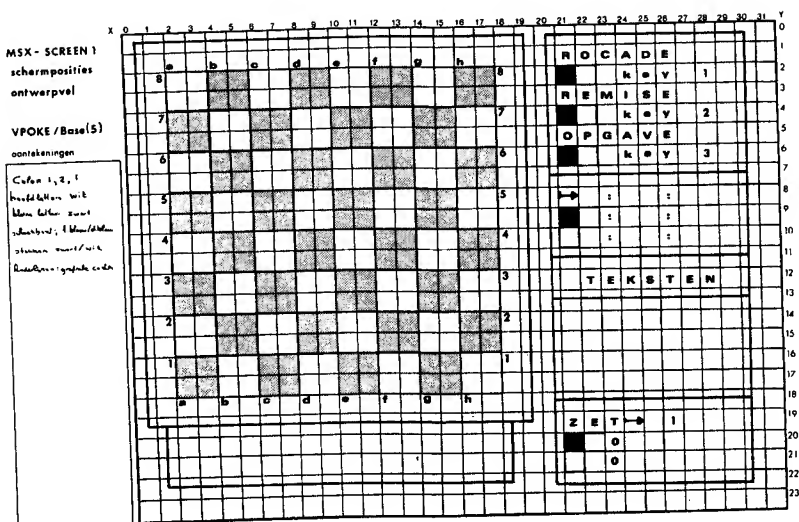
Natuurlijk willen we de zaken hier niet zo oppervlakkig aanpakken als het boven is gesteld. Al te zinvol is het nu ook weer niet, om aan een programma te beginnen, waarvan je je nog geen uitgediepte voorstelling hebt gemaakt. Het programma hier is dus dat van het schaakbord. U kunt daar weliswaar een programma aan overhouden, waarvan u plezier kunt hebben, maar het gaat eerst om iets anders natuurlijk. Hoe breng je het tot een programma. Al te simpel willen we het ons niet maken. Al zal ons programma het dan niet mogelijk maken om tegen de computer te schaken, we willen er evengoed de nodige zaken aan verbinden. Want al kunnen de 8 schaakstukken slechts naast elkaar gePRINT worden, evengoed willen we een stuk kunnen verplaatsen. Ermee over het schaakbord kunnen bewegen. Het moet bovendien allemaal via joystick en vuurknop te spelen zijn. Nu ja, bijna helemaal. Voor rokade, remise of schaakmat/opgave gaan we drie funktietoetsen gebruiken. Als echte schakers willen we ons binden aan een tijdlimiet. Bij overschrijding verliest de treuzelaar. Die tijd moet aldus afwisselend voor zwart en wit worden bijgehouden én met de gekozen tijd op het scherm te lezen zijn. En als zwart speelt, heeft wit z'n tijd af te wachten. Is dus ook in te bouwen. Laten we dan meteen ook maar onderaan het schaakbord weergeven welk stuk van welke positie naar welke positie vertrekt. En om de hoeveelste zet het gaat. Wat nog meer? Automatische promotie van een pion in een dame en vooral SCHAAK. Spelen we tegen onszelf, dan zouden we zo'n schaakpositie wel eens over het hoofd kunnen zien. Het moet overzichtelijk en plezierig schaken worden met dit scherm. Vooral met een partner. Eigenlijk prettiger dan met een schaakbord. En een verbetering moet het uiteraard hoe dan ook zijn.
Dat zullen we ons altijd ook voordien realiseren. Je kunt natuurlijk een geniaal programma opzetten, waarmee de uitkomst van de optelling 1+1 is weer te geven. De zin ervan zou evenwel omgekeerd evenredig zijn aan alle moeite. Wat we via de computer zullen realiseren moet voordelen kennen, welke het traditionele moet missen. Algemeen behoeft automatisering nog lang geen voordeel te zijn. Als we daardoor bijvoorbeeld het menselijke element in een handeling verliezen, dan zal dit tenslotte juist fnuikend gaan werken. Het kommerciële voordeel van de onderneming wordt weldra het sociale nadeel van de burgers. En aan vervreemding en vereenzaming ontbreekt het ons aller-

minst in de samenleving. Het mag dan zijn, dat geen tijd zoveel vooral elektronische kommunikatie mogelijkheden biedt, die vereenzaming houdt daarmee gelijke tred. We zullen daar ook een beetje rekening mee moeten houden bij het home-computeren. Bij voorkeur geen onwerkelijke spelletjes programmeren en tussendoor van mens tot mens blijven kommuniceren. Als het even kan al wandelend in de natuur. Want daarvan maken we deel uit en geenszins van dat elektronische. Het moest me even van het hart.

We willen dus behoorlijk wat met ons programma. Het zou nog kunnen worden uitgebreid met een volledige zetkontrole, zodat niet stiekum een stuk verkeerd zal worden gezet. Voor beginners een goede manier om de zetmogelijkheden van de schaakstukken te leren kennen. Maar in het programma zijn alleen enkele REM-regels ingelast. Daar kunt u eventueel zelf zo'n nogal uitgebreide routine onderbrengen. Want ook daar gaat het hier om. Om hier en daar lekker te knutselen met het programma. Tot nu toe hebt u het nodige aan regels moeten intikken en u bent er nog niet helemaal vanaf. We doen het voor de laatste maal wat anders. Omdat het mede om de opbouw van een programma gaat, zullen de regels telkens in kleine of grote plukken hier worden weergegeven. Op de volgorde van het intypen komt het niet aan. Het geeft vooral de gelegenheid om bij deze regels stil te staan. En behalve VPOKE en VPEEK, waarom het speciaal blijft gaan, zullen we zo ook de nodige BASIC-instrukties tegenkomen.

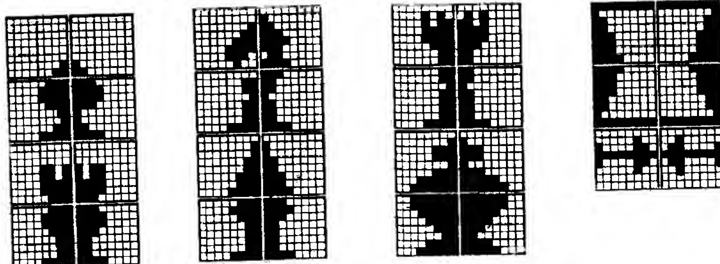
Op een schermvel, zoals in APPENDIX-V, tekenen we wat op het scherm dient te komen. Zo kan met het nodige passen en meten er de beste plaats voor gevonden worden. Zeker bij een vol scherm zoals hier moeten we zo te werk gaan. Waar komen de kaderlijnen, waar beginnen de vakken van het schaakbord en waar kunnen we tekst kwijt. Soms moeten we meerdere van die schermvellen gebruiken om het uiteindelijk allemaal op de beste plaats te krijgen. Zo netjes als het onderstaande schermvel hoeft het er dan natuurlijk niet uit te zien. Zolang het maar duidelijk blijft voor ons zelf en ermee gewerkt kan worden. En bij het opzetten van zo'n schermbeeld zal er rekening mee worden gehouden, dat regel 23 niet gePRINT kan worden.
Ook staat het schermbeeld veelal niet mooi in het midden. Vaak wijkt het naar rechts. Daarom zullen we in elk geval helemaal rechts geen tekst laten weergeven.

Het hoort evenzeer bij het programmeren. Helemaal onvoorstelbaar is het echt niet, dat u wellicht moeite hebt bij dit opzetten van een programma. Wie in het bezit komt van een computer, wordt min of meer voor de leeuwen geworpen. Ja, die Basic-instrukties zijn best te leren



met of zonder een cursus. Zo kun je ook leren zwemmen op het droge. Tenslotte gaat het er altijd maar om, wat je ervan terecht brengt, als je in het water wordt gegooid. Ofschoon bijvoorbeeld onze uitstekende MSX-magazines hieraan wel iets proberen te doen, zijn er daarin vaak veel te veel andere zaken, die uitgebreide aandacht nodig hebben ter informatie van de lezers. Zo is algemeen een begeleiding bij het opzetten van een programma ver te zoeken. Dit zal er mede de oorzaak van zijn, dat ook onze MSX weldra tussen de motteballen terecht komt. Of alleen voor gekochte spelletjes wordt benut. En daarvoor werd hij niet gekocht. We proberen er hier iets aan te doen.

Hebben we dat schermbeeld zover voorbereid op het schermvel, dan kunnen we ons bezig houden met speciaal aan te maken figuren. Hier



dus de schaakstukken. Met PRENT vlot te ontwerpen. Dan is er nog een aanwijssprite nodig, omdat we met afstandbediening willen werken. En een paar pijltjes.

Zo ziet het resultaat van onze ontwerp drift eruit en de DATA-regels hebben we genoteerd en bij het ontwerpvel gevoegd. Genoteerd zijn vanaf dit schermvel ook de X en Y waarden, die we aan de LOCATE's hebben te geven om het allemaal daar te krijgen, waar we het willen hebben. Het kan ook direkt van het scherm-ontwerpvel worden afgelezen.

Nu lijkt het wat gemakkelijker dan het zal blijken te zijn. We hebben niet alleen met zwarte en witte stukken te maken. Beide soorten moeten ook de achtergrondkleur hebben van de vakken. Transparant als achtergrondkleur kan hier niet. De vakken van het bord bestaan uit kleurblokjes en die kleur zou dan gewist worden. Voor de donkerblauwe is dat geen ramp, omdat dit ook de schermkleur is. Zo zijn er voor elk schaakstuk 4 sets nodig, die vooraf in de juiste kleurenkombinatie dienen te worden gezet. Wél 24 karakters voor 1 stel schaakstukken, maar 4x24 om alle PRINT-variaties mogelijk te maken. Tegelijk willen we een schaakstuk ook als sprite gebruiken om het over het bord te kunnen bewegen. Extra DATA-regels zijn daar niet voor nodig en met de kleuren hoeft geen rekening te worden gehouden. We moeten aldus nog een lijstje maken om te kunnen noteren welke kodes voor welke PRINT-variatie zulen worden gebruikt. De setlijsten van APPENDIX-1 zijn hiervoor te raadplegen. Die houden we meteen bij de hand om na te kunnen gaan, welke toetsen in te drukken zijn om de gekozen karakter in een PRINT te kunnen zetten. Veranderd in het betreffende schaakstuk. Als we dit allemaal bijeen hebben vergaard, dan beginnen we aan het intikken van ons programma. En voor het prille begin zijn er aldus deze regels:

```
10 REM ** SCHAAKBORD/JOYSTICK/CLOAD"SCHA
AK"/ progr.nr.10 Basis van BASIC / Ca 10
100 bytes
20 CLS:SCREEN1,2:WIDTH 32:COLOR 1,2,6:KE
Y OFF:X$=SPACE$(9)
30 INPUT"informatie? J/N";A$:IF A$="J" T
HEN GOSUB 2290 ELSE 40
40 INPUT"speeltijd in uur/minuten";U1,M1
50 IF U1>9 OR M1>59 OR M1<1 THEN 40
60 CLS:GOSUB 2610:GOSUB 2780
```

De REM-regel waarmee gestart wordt, zal altijd een programma openen. U kunt daarin behalve de naam en eventueel het volgnummer van uw programma b.v. de datum en uw naam/woonplaats vermelden. Want ú bent de auteur van uw programma. Daaraan zijn bij de wet geregelde auteursrechten verbonden. Immers, u hebt er uw ideetjes en vooral uw tijd ingestoken. Dat mag één ander niet zondermeer van u overnemen. Zeker, in de praktijk blijven de meeste programma's binnenshuis. Wellicht zal een ander ze nooit te zien krijgen. Het is echter een goede zaak, wanneer we ons van dit auteursrecht bewust blijven. Want dan zullen we ons wel tweemaal bedenken, alvorens het programma van een ander klakkeloos te kopiëren.

Regel 20 spreekt voor zichzelf. Meestal wordt hier en/of in de volgende regel de aan de te gebruiken variabelen toe te kennen startwaarden vermeld. Hier is het beperkt tot X$. Het is de enige algemene variabele, zoals in het programma te gebruiken voor het wissen van een regel. Uiteraard niet de enige variabele. Daarvan zullen we echter de startwaarden even vaak nodig hebben, als er partijen schaak gespeeld zullen worden. Daarom is zo'n variabelenregel beter als startregel te gebruiken en hier aldus lager te zetten. Vooral in programma's waarin allerhande akties mogelijk zijn, mag het niet aan informatie onbreken. Het wordt nogal eens verwaarloosd. Dan moet je maar raden, welke toetsen er in te drukken zijn en wat er al dan niet kan met het programma. De programma-maker weet er alles van en een INPUT te beantwoorden met RETURN kan voor hem amper storend zijn. Maar krijgen we na lange tijd weer eens een eigen programma onder ogen, dan zullen we evenmin nog precies weten van de hoed en de rand. Overigens is ook INPUT met LOCATE op een gewenste schermpositie te zetten, indien dit nodig mocht zijn. In dit geval bestaat daarvoor geen noodzaak. Hoewel aan INPUT een tekst kan worden meegegeven, behoeven we ons niet altijd van deze instruktie afhankelijk te maken. We kunnen ook de speciale toetsen van de computer gebruiken in bijvoorbeeld een INKEY$-instruktie.
Daarbij kan het volgende lijstje u van dienst zijn:

Speciale toetsen

code &H	code dec.	toets(en)
03	3	STOP/CTRL
08	8	BS
09	9	TAB
0B	11	HOME

0C	12	CLS (HOME&SHIFT)
0D	13	RETURN
12	18	INS
18	24	SELECT
1B	27	ESC
1C	28	CURSOR rechts
1D	29	CURSOR links
1E	30	CURSOR omhoog
1F	31	CURSOR omlaag
7F	127	DEL

Ze zijn op de volgende manier te gebruiken:

```
A$=INKEY$:IF  A$=CHR$ (13) THEN . . .
```

Vanzelfsprekend geven we in een tekstregel wel even aan, dat zoals in dit voorbeeld de RETURN-toets ingedrukt dient te worden. Wanneer er eigenlijk geen noodzaak bestaat, om bijvoorbeeld toets J of K in te laten drukken, dan is het beter van de speciale toetsen gebruik te maken. Ze zijn er tenslotte voor. Al zullen we in de praktijk daarvan maar een enkele gebruiken.

Voordat we de regels van de informatie-routine hier gaan weergeven, zodat ze door u ingetikt kunnen worden, eerst nog even regel 40 en 50. De INPUT vraagt om de speeltijd. Weer te geven in aantal uren en minuten gescheiden door een komma. Aan deze tijd bindt de gebruiker zich. Ook als meerdere partijen achtereen gespeeld zullen worden, blijft de eenmaal aangegeven speeltijd gehandhaafd. Er MOET een tijd worden ingetikt. In regel 50 is daaraan echter een voorwaarde te verbinden. De variabele U1 voor de uren mag geen hogere waarde dan 9 krijgen. Behalve dat dit toch al een fors tijdlimiet is, is er op het scherm simpelweg geen plaats voor 2 cijfers bij uren. Verder mag aan M1 voor minuten natuurlijk geen hogere waarde dan 59 worden gegeven en verplicht M1< 1 tot in elk geval de INPUT van een tijdlimiet. Bij een verkeerde input komt regel 40 terug op het scherm. Aan IF . . . THEN regels valt niet te ontkomen. Het is absoluut geen schande om deze instruktie veelvuldig te gebruiken in een programma. Die indruk wordt soms wél gewekt. Vooral bij de opbouw van ons programma, zullen we er kwistig gebruik van maken. Loopt het allemaal eenmaal gesmeerd, dan kunnen we alsnog nagaan in hoeverre IF . . . THEN regels te vervangen zijn door een kompakte formulering. Vaak is dat

heel goed mogelijk en kunnen voorwaarderegels vervangen worden door één enkele regel.
Computeraars die goed zijn in toegepaste wiskunde hebben hier een flinke voorsprong. Voor de minder begaafden is het vaak een kwestie van proberen, alvorens ook zij een flink aantal voorwaarderegels weten te vervangen door een enkele regel. Daarin zit trouwens ook een uitdaging. Met zo weinig mogelijk regels zoveel mogelijk te laten uitvoeren. Dat is vooral dat efficiënte programmeren. Echter ook hier zal de duidelijkheid het eerst langzaam maar zeker afleggen tegen het efficiënte. Alles op z'n tijd.

Het mag zijn, dat we veel instrukties kwijt kunnen in één regel, maar evenzeer bij MSX kan de IF . . . THEN instruktie hierop een uitzondering vormen. Staat deze aan het begin van de regel, dan is het doorgaans weinig zinvol, om daarachter nog andere instrukties te zetten. Want als aan de te stellen voorwaarde niet wordt voldaan, dan wordt die regel eenvoudig niet verder gelezen en naar de volgende gesprongen. Na een IF . . . THEN kan uitsluitend vermeld worden, wat er gebeuren moet, indien wél aan de voorwaarde wordt voldaan. Maar dat wist u allicht reeds. In regel 20 is onder andere WIDTH 32 opgenomen. Screen 1 heeft een schermbreedte van 28 kolommen en we willen graag de hele schermbreedte benutten. Ook om naderhand in de adressen van de schermlokaties met deze breedte van 32 te kunnen werken. U bent eerst weer aan wat typwerk toe. Onze informatie-routine.

```
2290 REM ** informatie **
2300 CLS
2310 PRINT"      ***** SCHAAKBORD *****"
2320 LOCATE 5,1:PRINT STRING$(22,195)
2330 PRINT"1) Zetten door aanwijzing via
      joystick"
2340 PRINT"2) Automatische promotie van
      pion in dame"
2350 PRINT"3) Controle op verkeerde zet
en    schaak"
2360 PRINT"4) Bepaling tijdlimiet"
2370 PRINT"5) Weergave zet, beurt, score
 en   tijd"
2380 PRINT"6) Funtietoetsen:"
2390 PRINT
2400 PRINT"   TOETS 1: rocade; eerst tor
```

```
en-   zet, dan toets 1 en koningzet"
2410 PRINT"   TOETS 2: remise;J-TOETS vo
ort   zetting met zelfde tijdlimiet   N-
TOETS einde spel"
2420 PRINT"   TOETS 3: opgave; ook bij
       schaakmat. Via J of N-toets     vo
ortzetting of einde"
2430 PRINT
2440 RETURN
```

Het spreekt voor zichzelf. Alleen regel 2320 pikken we er nog even tussen uit. Voor het onderstrepen van woorden is karakter 195 goed te gebruiken. Hier kan dit nog, want weldra gaan we deze karakter een andere vorm geven.
Er kan gebruik worden gemaakt van de STRING$-instruktie. Het tweede getal tussen de haakjes is aldus kodenummer 195 en het eerste geeft aan hoe vaak dit karakter moet worden weergegeven. Zo krijgen we een nette streep onder de PRINT van regel 2310. En u weet natuurlijk, dat PRINT zonder enige bijvoeging goed is voor een lege regel tussen bijvoorbeeld twee PRINT-regels. Maar er dient hier nog wel tenminste 1 schermregel over te blijven. Via de RETURN komt de INPUT van de speeltijd op het scherm. Als ook hieraan is voldaan, wist de CLS in regel 60 het scherm en kan worden begonnen aan de opbouw van het schermbeeld. Bij meerdere PRINT-regels zoals in de informatie routine lukt het doorgaans niet meteen, om zinnen netjes op het scherm te krijgen. Woorden moeten worden afgebroken en soms is er met extra spaties te werken of moeten twee woorden juist aaneen worden getypt. Het vergt de nodige ingrepen. Het mag evenwel een goede gewoonte worden, om ook uitgebreidere teksten netjes op het scherm te zetten. Dat hoort gewoon bij de goede verzorging van het programma. Zo gebeurt het echter lang niet altijd.

Regel 60 kent twee GOSUBS. Ze verwijzen beide naar de routine, waarin het schermbeeld wordt opgebouwd. Er hoort nog een derde GOSUB bij. Normaal zal een enkele GOSUB toereikend zijn om deze routine aan het werk te zetten. Immers, een schermbeeld bouwen we algemeen slechts één keer op en daar doen we het dan mee. Anders wordt dit echter als een spel, zoals hier herhaald gespeeld kan worden zonder dat hiervoor het programma behoeft te worden afgebroken. Dán moet die schermopbouw, althans ten dele, herhaald kunnen worden. Om die reden is de routine in drie parten verdeeld, afgesloten door een RETURN. U mag het eerste partje intypen.

```
2610 REM ** kleur/chars **
2620 LOCATE 10,10:PRINT"even geduld":LOC
ATE 13,12:PRINT"a.u.b."
2630 B=BASE(6)
2640 FOR C=16 TO 31:VPOKE B+(C/8),&H12:N
EXT C
2650 FOR C=128 TO 151:VPOKE B+(C/8),&H15
:NEXT C:FOR C=152 TO 175:VPOKE B+(C/8),&
H14:NEXT C
2660 FOR C=176 TO 199:VPOKE B+(C/8),&HF5
:NEXT C:FOR C=200 TO 223:VPOKE B+(C/8),&
HF4:NEXT C
2670 FOR C=48 TO 63:VPOKE B+(C/8),&H12:N
EXT C:FOR C=97 TO 122:VPOKE B+(C/8),&H12
:NEXT C:FOR C=65 TO 95:VPOKE B+(C/8),&HF
2:NEXT C
2680 FOR C=224 TO 231:VPOKE B+(C/8),&H55
:NEXT C:FOR C=232 TO 239:VPOKE B+(C/8),&
H44:NEXT C
2690 FOR C=240 TO 247:VPOKE B+(C/8),&HFF
:NEXT C:FOR C=248 TO 255:VPOKE B+(C/8),&
H11:NEXT C
2700 REM ** schaakstukken **
2710 A=BASE(7)+1024:G=0
2720 RESTORE 2840
2730 FOR I=0 TO 191:READ C$:VPOKE A,VAL(
"&H"+C$):A=A+1:NEXT I
2740 G=G+1:IF G<4 THEN 2720
2750 G=0:A=BASE(7)+728:RESTORE 2910
2760 FOR I=0 TO 15:READ C$:VPOKE A+I,VAL
("&H"+C$):NEXT I
2770 RETURN
```

De DATA-regels die horen bij de regels 2700 t/m 2700 komen nog. Zoveel zijn het er overigens niet. Dergelijke DATA-regels kunnen welis-waar overal in een programma staan, maar algemeen zijn ze toch beter bijeen en aan het eind van een routine te plaatsen. Dat in deze regels RESTORE wordt gebruikt heeft aldus te maken met een herhaald

gebruik van de routine. Zo kunnen de DATA-regels desgewenst nog eens gelezen worden. In dit programma zal het niet nodig zijn. Evenmin zou de tweede GOSUB nodig zijn geweest. Deze betreft de sprites. Het is evengoed toch zo gesplitst voor het geval er bij het herhaald spelen ongewenste sprites op het scherm aanwezeg zouden blijven. Dit is mogelijk, als bijvoorbeeld niet goed op een vakje wordt gezet. Oorspronkelijk was daarvoor in regel 80 DEFUSR=&H69:A=USR (0) opgenomen met daarvoor aldus de tweede GOSUB naar de sprite-routine. Het gebruik van dit middel om sprites te verjagen is evenwel dusdanig radikaal, dat die sprites opnieuw geVPOKEd moesten worden in de adressen van BASE (9). Dat werkt natuurlijk vertragend bij de heropbouw van het schermbeeld. En omdat de genoemde situatie zich nauwelijks voordoet, is het dus maar weggelaten.

Wie meer ervaring heeft met programmeren, zal wellicht dit kommentaar bij de opbouw van het schaakbordprogramma minder kunnen waarderen. Persoonlijk evenwel leer ik altijd wel weer wat bij het analyseren van andermans programma's. Al was het maar van de manier waarop één en ander tot stand wordt gebracht. En eens op herhaling gaan, houdt ook het nederlandse leger paraat. De hele kwestie is eenvoudig, dat zulk kommentaar alom ontbreekt. Voor beginners of voor hen met nog weinig programmeerervaring is dat een groot nadeel. Al is het ene programma dan het andere nog niet, de opbouwstruktuur kent veel overeenkomsten. En het is bovenal van belang om bij die opbouw met van alles rekening te houden. De meest onnozele fout is veelal het lastigste te ontdekken. Het bederft het plezier in het programmeren soms zodanig, dat we toch maar liever beter leren voetballen. Maar nog belangrijker dan eigen, goed werkende programma's, is de vertrouwdheid met dit programmeren. Het neemt het geheimzinnige van automatiseringen weg, die slechts kunnen funktioneren via een programma, hoezeer ook ingebakken in de chips. Een volslagen computer-analfabeet te zijn is in onze dagen niet meer zo raadzaam.
In regel 2620 laten we "even geduld" op het scherm zetten. Al dat VPOKEn kost nu eenmaal wat tijd. En een niet ingewijde mocht eens menen, dat er helemaal niets gebeurt. Intussen zijn we echter aangekomen bij waarom het hier speciaal gaat; dat VPOKEn. We beginnen met de kleurverandering van de te gebruiken sets. En dat zijn er heel wat. Bij set 2 en 3 gaat het om die kaderlijntjes, waarvan de achtergrondkleur aangepast moet worden bij de schermkleur. We kunnen uiteraard zulke lijntjes ook zelf ontwerpen, maar vanwaar die moeite als de bestaande goed te gebruiken zijn? En hier kan dat. U bent intussen al zodanig vertrouwd geworden met dit VPOKEn van de kleuren, dat we daarop niet meer in hoeven te gaan. Al hebben we maar zes schaakstukken nodig, voor elk stuk moeten wel vier karakterkodes

worden aangesproken. Dat zijn er aldus totaal 24 en zoveel zitten er net in 3 sets. Omdat bij het begin beginnen doorgaans een goede gewoonte is, doen we dat hier ook in regel 2650. De kodes van de sets 17 t/m 19 gaan van 128 tot en met 151. En omdat u intussen het gemak van de hex-kode ook hebt ontdekt, ziet u meteen, dat het hierbij gaat om de zwarte stukken met een lichtblauwe achtergrond. Dezelfde hebben we ook nodig met een donkerblauwe achtergrond. Daarvoor aldus de sets 20 t/m 22 gebruikt. In regel 2660 doen we hetzelfde voor de witte stukken. Dat tikt al lekker aan. Alleen van kleurveranderen doen we de sets 7 en 8 (zwarte cijfers op de schermkleur), 13 t/m 16 (kleine letters in dezelfde kleurenkombinatie) en de sets 9 t/m 12 (grote letter in wit op de schermkleur). U kunt nu terecht opmerken, hetgeen geheel de bedoeling is, dat bijvoorbeeld de kleine lettersets noch met het eerste kodenummer zijn begonnen, noch daarmee zijn geëindigd. Dat dit geen noodzaak is, om toch deze sets de gewenste kleur te geven, zal evenwel de goede gewoonte niet achterhalen. Want wat niet echt moet, behoeft niet goed te zijn. Voor ons eigen overzicht blijft het beter om voor een set bij het eerste kodenummer te beginnen en bij de laatste te eindigen. Dan zal er geen misverstand ontstaan.

Opnieuw 2 hele sets zijn nodig voor de kleurblokjes lichtblauw en donkerblauw, waarmee de vakjes van het schaakbord opgebouwd moeten worden. Er is hier gekozen voor de simpele manier door vóór en achtergrondkleur gelijk te maken. Alleen als we spaarzaam met de sets moeten omgaan, is het verstandiger om aldus zo'n vol blokje met 8 x &HFF te VPOKEn in een karakter. Dan blijven de overige karakters van zo'n set nog bruikbaar. We hebben die kleurblokjes ook nodig, omdat daarmee een karakterkode wordt verbonden. Kode 224 staat voortaan voor het lichtblauwe en kode 232 voor het donkerblauwe vakje zonder een schaakstuk. Dit houvast zal nodig zijn, als we via VPEEK het scherm laten nazoeken op aanwezige kodes. De laatste kleurwijziging treft meteen ook de beide laatste sets.
We willen nog een zwart en een wit blokje hebben ter vervanging van de tekst "zwart" en "wit". Dit omdat er niet al te veel ruimte is op het scherm. Zo kan het ook. Wat we aan kleuren nodig hebben, is hiermee alvast geVPOKEd in BASE (6). Nu alleen nog de benodigde karakters veranderen in onder andere de schaakstukken. Dat betekent weer wat typewerk voor u.

```
2780 REM ** stukken-sprites **
2790 A=BASE(9):G=0:RESTORE 2840
2800 FOR I=0 TO 191:READ C$:VPOKE A,VAL(
```

```
"&H"+C$):A=A+1:NEXT I
2810 RESTORE 2900
2820 FOR I=0 TO 31:READ C$:VPOKE A+I,VAL
("&H"+C$):NEXT I
2830 RETURN
2840 DATA 00,00,00,00,00,00,01,03,07,0F,
0F,07,01,01,03,0F,00,00,00,00,00,00,80,C
0,E0,F0,F0,E0,80,80,C0,F0 ' pion
2850 DATA 00,00,00,0D,0D,0F,0F,07,07,0F,
0F,07,03,03,03,0F,00,00,00,B0,B0,F0,F0,E
0,E0,F0,F0,E0,C0,C0,C0,F0 ' toren
2860 DATA 00,00,01,03,07,0F,0E,04,00,03,
03,01,03,03,03,0F,00,80,C0,E0,F0,E0,60,F
0,E0,C0,C0,80,C0,C0,C0,F0 ' paard
2870 DATA 00,01,01,03,03,07,07,0F,07,0F,
07,03,03,03,03,0F,00,80,80,C0,C0,E0,E0,F
0,E0,F0,E0,C0,C0,C0,C0,F0 ' loper
2880 DATA 00,0A,0B,0F,0F,07,03,03,03,03,
01,03,03,03,03,0F,00,50,D0,F0,F0,E0,C0,C
0,C0,C0,80,C0,C0,C0,C0,F0 ' dame
2890 DATA 00,01,07,07,01,07,1F,3F,3F,3F,
1F,0F,07,03,0F,1F,00,80,E0,E0,80,E0,F8,F
C,FC,FC,F8,F0,E0,C0,F0,F8 ' koning
2900 DATA FF,80,C0,C0,E0,E0,E0,F0,F0,E0,
E0,E0,C0,C0,80,FF,FF,01,03,03,07,07,07,0
F,0F,07,07,07,03,03,01,FF  ' aanwijsspri
te
2910 DATA 00,04,86,FF,86,04,00,00 ' pijl
tje 1
2920 DATA 00,20,61,FF,61,20,00,00 ' pijl
tje 2
```

Valt wel mee met die DATA-regels, nietwaar? Er is met behulp van de apostrof bij vermeld om welk ontwerp het gaat. Mocht er wat gewijzigd moeten worden aan de vorm van een schaakstuk, dan is de betreffende DATA-regel meteen te vinden. En allicht wilt u van de koning een minder dikbuikig model maken in het geval u een warm aanhanger van de monarchie bent. Nu, dat is dus te doen in regel 2890. Echter worden dezelfde DATA-regels ook benut voor de sprite-patronen, zoals te

VPOKEn in de adressen van BASE(9). Vandaar opnieuw RESTORE. We hebben inklusief de aanwijssprite 7 van deze dingen nodig, samengesteld uit een matrix van 16x16. Het vergt aldus totaal 192 adressen van BASE (9). Eigenlijk was het niet nodig om voor die aanwijssprite een aparte VPOKE-regel te bedenken. Het is gedaan om de schaakstuksprites apart te houden. Want de volgorde waarin zij in de adressen gezet worden, is bepalend voor het spritenummer en daarmee krijgen we nog te maken.
Al kunnen er dan geen 8 sprites als schaakstukken naast elkaar, met zo'n sprite is een schaakstuk wél van de ene naar de andere positie op het schaakbord te bewegen. Dat kunnen we doen door het gePRINTe schaakstuk te wissen in de kleur van het betreffende vakje en daarvoor in de plaats een sprite te laten opdraven. Dit gebeuren wordt dan alleen herkenbaar, als we opeens met joystick of cursortoetsen het schaakstuk over het speelveld kunnen bewegen. De sprite dient ons aldus slechts als zo'n hulpmiddel.

De laatste VPOKE-aktie is gemeend voor 2 pijltjes, die slechts een ondersteunende funktie zullen krijgen. De patroonwaarden hadden natuurlijk in één DATA-regel ondergebracht kunnen worden. Bij het opzetten van een programma zullen we aanvankelijk met schaars bezette regels werken. Immers, we willen graag het overzicht behouden. Naderhand kunnen dan altijd nog ook meerdere onnodige korte regels aaneen worden gebreid. Dat is min of meer letterlijk mogelijk met MSX. Wilt u bijvoorbeeld regel 2920 onderbrengen in de voorgaande, dan zet u gewoon de cursor direkt achter de laatste waarde van regel 2910; aldus 00. U typt een komma en daarna net zoveel willekeurige tekens, bijvoorbeeld de X, tot ze aansluiten op de eerste 00 van regel 2920. Zo komen beide regels aan elkaar vast te zitten. Even die regels listen en met de DEL-toets al die X-tekens weghalen. Zo staat het keurig op één regel en u kunt regel 2920 verwijderen. Ofschoon dit veelal erg handig is, kent het over het algemeen ook een nadeel. Wilt u bijvoorbeeld iets aan een regel toevoegen, dan komt als gebruikelijk na het LISTen Ok onder die regel te staan. Of Break. We moeten oppassen, dat zoiets niet aan onze regel gaat plakken. Dat kan gemakkelijk, als we zonder de INS-toets te gebruiken nog net iets aan de regel denken te kunnen toevoegen. Als we daarmee evenwel de laatste positie van de schermbreedte hebben bereikt, dan zit alles wat op de volgende positie staat daar wel mooi aan vast. In geval van twijfel is het altijd wijs om dit even te kontroleren. Anders kunnen we bij het RUNnen een SYNTAX error vermeldt krijgen en ons de ogen tranend speuren naar de oorzaak.

In regel 2790 is de tel-variabele G weer op 0 gezet. Het is gebruikt om

dezelfde DATA-regels 4 maal te laten lezen met de verwijzing naar RESTORE. Het gebruik van tellers komt veel voor in programma's. Het kan echter de oorzaak zijn van zo'n onnozel foutje. Na gebruik meteen weer op 0 zetten mag een goede gewoonte zijn. Over dat gebruik van variabelen kunnen we het nu mooi even hebben. Het maakt vooral de kracht uit van onze computer. Echter ook in dat rijk der variabelen zullen we met enig beleid aan de slag gaan. Eigenlijk hoort dit ook bij standaardisering. Het maakt zowel voor ons zelf als voor anderen het programma leesbaarder. Alle mogelijke letters, letterkombinaties en kombinaties met cijfers kunnen worden gebruikt.
Daarvoor zullen we het alfabet echter niet in een hoge hoed stoppen om er na enig schudden een willekeurige letter uit te halen. Variabelen moeten zo herkenbaar mogelijk blijven en als het even kan toch beperkt blijven tot een enkele letter. Voor lussen wordt veelal de I en/of de J gebruikt, behalve als het om een speciale lus gaat. Zoals bijvoorbeeld bij het VPOKEn van de kleuren. De C herinnert dan aan de kodes, die we erbij gebruiken. Variabelen zijn ruwweg in 3 kategorieën te verdelen. De bij herhaling gebruikte, de incidenteel gebruikte en de algemene. De laatste dient meestal als een hulpmiddel, al dan niet herhaald, en daarvoor is desnoods van alles te gebruiken. Tot de incidentelen behoren dus die lusvariabelen. Of tellers. We kunnen er de G of de F voor gebruiken. Waar we dan zo'n variabele tegenkomen, daar is die dan meteen als teller te herkennen. Doorlopende telwaarden horen bij de herhaald te gebruiken variabelen. Daarvoor kiezen we in elk geval geen willekeurige letters. Het gebruik van X en Y voor de coördinaten in bijvoorbeeld sprites of LOCATE blijft duidelijk maken, dat X voor kolom en Y voor regel staat. Het is aan te vullen met K en R of met V(vertikaal) en H(horizontaal) als er sprake is van meerdere variabelen voor schermposities. De Q en de M zeggen hiervoor zo weinig. Dat geldt voor alle herhaald te gebruiken variabelen. We moeten er vooral geen rommelpot van maken. Vaak kan blijken, dat we eigenlijk niet eens zo gek veel variabelen nodig hebben, omdat we bijvoorbeeld variabelen met dezelfde betekenis hebben voorzien van een cijfer. Maar al zou dit strikt genomen niet mogen gebeuren, het kán wel plaatsvinden, dat bijvoorbeeld twee bijna identieke variabelen verkeerd worden gelezen door de computer. Wanneer zulke variabelen in achtereenvolgens te verwerken instrukties worden gebruikt, dan kan het er de oorzaak van zijn, dat we verkeerde resultaten krijgen. In deze gevallen moet voor duidelijk van elkaar te onderscheiden variabelen worden gekozen. Wilt u er eens een stelletje intypen?

```
70 REM ** start/herstart **
80 GOSUB 2930
```

```
90 X1=0:S1=0:M2=0:U2=0:X2=0:S2=0:M3=0:U3
=0:ZET=0:SP=1:N=0:XP=522:PX=74:X=16:Y=15
9:E=0:F=0:G=0
```

Regel 70 is de eigenlijke start van het programma. En de herstart als we nog een partij schaak willen spelen. Omdat daarvoor het scherm opnieuw gePRINT moet worden, komen we eerst in regel 80 de GOSUB tegen, die deze routine in aktie laat komen. En dat gaat zo snel, dat we in die tijd amper het woord computer kunnen spellen. Eerst even de gebruikte variabelen. Bij de INPUT van de speeltijd zijn al de variabelen U1 en M1 gebruikt. In regel 90 komen we de familieleden tegen, die dienst zullen doen in de klokroutine, waarmee de speeltijd voor zwart en wit is bij te houden; M2, U2, M3 en U3. Daarbij horen X1, S1, X2 en S2. Dat de S hier voor sekonde staat behoeft geen nader betoog. X1 en X2 zijn hulpjes bij de tijdmeting.
We komen de hele familie naderhand nog tegen. Als variabele lijkt ZET aan de lange kant. Maar de Z met identieke betekenis wordt in het programma gebruikt en veelvuldiger dan ZET. Deze Z en nog wat andere variabelen hebben geen startwaarde nodig. Ze zullen hun wisselende waarde in de routines ontvangen. De variabele SP is een belangrijke. Het staat voor SPeler en hier is SP=1 wit met naderhand SP=0 voor zwart. Het zal in routines nodig zijn om tussen wit en zwart te onderscheiden. De N, E, F en G zijn als tellers gebruikt en moeten aldus de 0 als startwaarde hebben. In XP en PX is weer een schermpositie te herkennen. Van de witte en de zwarte koning. Deze waarden zullen we hard nodig hebben om de SCHAAK-situaties te kunnen vaststellen. Tenslotte zijn de als zodanig herkenbare X en Y de startwaarden van de aanwijssprite. Bij een volgende partij komt die weer op deze plaats te staan. Wat er verder nog aan variabelen wordt gebruikt, dat komen we vanzelf tegen. Het belangrijkste blijft hier, dat we die herhaald te gebruiken variabelen tenminste zelf blijven herkennen.

De GOSUB in regel 80 moet type-aktie teweeg brengen. Dat zal ditmaal nog niet zo simpel zijn. U moet er APPENDIX-I voor raadplegen om via de juiste toetsen het bewuste karakter op het scherm te krijgen in de PRINT-regels. En wat u daarmee zult intypen, is het gehele schermbeeld inklusief het schaakbord met de stukken. Al zie je dat er niet aan af. Maar we hebben die karakters immers veranderd in de eigen ontwerpen en kleuren. Om het u wat gemakkelijker te maken het betreffende karakter te herkennen, kan hier de tweede methode van stal worden gehaald. Dat is dus het direkt VPOKEn van karakterkodes in de schermadressen van BASE (5). Daarvoor zijn de kodenummers van die karakters in DATA-regels onder te brengen. Maar u kunt het

overtypen daarvan besparen. Ze worden alleen afgedrukt om u duidelijker te maken om welke karakters het gaat. U behoeft het kodenummer alleen op te zoeken in APPENDIX-1 en in de PRINT-regels te typen volgens de aangegeven toetskombinatie. Ofschoon dit VPOKEn in BASE (5) nog sneller het schermbeeld doet verschijnen, is toch voor het PRINTen gekozen. Wat nu nog lijkt op vooral grieks geheimschrift, dat openbaart zich na een eerste RUN helemaal als het schaakbord en de bijbehorende tekstkaders. Zo krijgen we het hele schermbeeld in PRINT-regels voor ons en is het veel gemakkelijker om korrekties aan te brengen. Wanneer het hele programma is ingetikt en u hebt geRUNd, dan moet u deze regels nog maar eens bekijken. Mocht u dan toch een verkeerde karakterkode hebben ingetikt, dan is meteen te zien waar zich deze tragedie heeft voltrokken. Vooral dankzij deze regels zult u herkennen, van hoeveel belang het is om een karakter zonder tijdrovend zoeken met de juiste toetsen op het scherm te kunnen zetten. Natuurlijk kan het ook via CHR$. Maar dan wordt het allemaal net zo langdradig als bij de SPRITE$ instruktie.

```
2930 REM ** scherm opmaak **
2940 LOCATE 0,0:PRINT"┌──────────
──────────────┐┌──────────
┐"
2950 PRINT"| a b c d e f g h  ||ROCAD
E   |"
2960 PRINT"|8äàáóîÄ¿¬öûñªêè£₧8||°≡ ke
y 1 |"
2970 PRINT"| àçíúìÁ⌐½òùÑºëï¥ƒ ||REMIS
E   |"
2980 PRINT"|7ÿüÇéÿüÇéÿüÇéÿüÇé7||°≡ ke
y 2 |"
2990 PRINT"| ö¢üåö¢üåö¢üåö¢üå ||OPGAV
E   |"
3000 PRINT"|6αα♀♀αα♀♀αα♀♀αα♀♀6||°≡ ke
y 3 |"
3010 PRINT"| αα♀♀αα♀♀αα♀♀αα♀♀ ||├──
 ──────────┤"
3020 PRINT"|5♀♀αα♀♀αα♀♀αα♀♀αα5||[
 : 0|"
3030 PRINT"| ♀♀αα♀♀αα♀♀αα♀♀αα ||°
     |"
```

```
3040 PRINT"|4αα⧱⧱αα⧱⧱αα⧱⧱αα⧱⧱4||≡
     |"
3050 PRINT"| αα⧱⧱αα⧱⧱αα⧱⧱αα⧱⧱ ||├──
──────────┤"
3060 PRINT"|3⧱⧱αα⧱⧱αα⧱⧱αα⧱⧱αα3|| TEKS
TEN |"
3070 PRINT"| ⧱⧱αα⧱⧱αα⧱⧱αα⧱⧱αα ||├──
──────────┤"
3080 PRINT"|2[illegible]2||
     |"
3090 PRINT"| [illegible] ||
     |"
3100 PRINT"||1[illegible]1||
     |"
3110 PRINT"| [illegible] ||
     |"
3120 PRINT"| a b c d e f g h ||├──
──────────┤"
3130 PRINT"└┬──────────────────────
───┴ |ZET[      |"
3140 PRINT" |                  | |°
     |"
3150 PRINT" |                  | |≡
     |"
3160 PRINT" └──────────────────────
───┘  └──────────────┘"
3170 LOCATE 22,20:PRINT ZS:LOCATE 22,21:
PRINT WS:LOCATE 22,8:PRINT USING"##:##";
U1,M1:LOCATE 25,19:PRINT ZET
3180 RETURN
```

In regel 2940 is slechts éénmaal LOCATE gebruikt. Door de schermpositie van de eerste PRINT-regel zo te bepalen, sluiten alle andere PRINT-regels zich daarop aan. Hierdoor kunnen we ons de LOCATE's voor de overige PRINT's besparen. Uiteraard is het geen noodzaak om de DATA-regels hieronder te raadplegen bij het intikken van de karakters. Als u ze zo ook kunt vinden, dan scheelt dat weer wat aan tijd. In elk geval weet u dan evengoed hoe de karakters ook direkt te VPOKEn zijn.

```
A=BASE (5)
FOR I=0 TO 703
READ C
VPOKE A+I, C
NEXT I
```

Alle DATA-regels eindigen met spatiekode 32, omdat de laatste schermkolom niet is gebruikt. Vanzelfsprekend behoeft een PRINT-regel niet met een spatie te worden afgesloten. In de op wat minder gebruikelijke wijze weergegeven DATA-regels zijn alle ASCII-kodes van 32 t/m 127 *kursief* getypt. Het gaat daarbij aldus om de standaardkarakters, veelal de spatie, die u ook zo van het toetsenbord kunt intikken. Bij het VPOKEn moeten alle karakterkodes in de DATA-regels aanwezig zijn. Aan de hand hiervan is snel in APPENDIX-1 op te zoeken welke toetsen in te drukken zijn om het betreffende karakter in de PRINT-regel te zetten. Het lijkt lastiger dan het is. De uitzondering is kodenummer 201, zoals gebruikt in regel 3090. Hiervoor is geen toetsenkombinatie aangegeven. Het kan een lege matrix zijn of een smalle vertikale streep geheel rechts. Waarschijnlijk kunt u de kode intikken met:

Caps-Lock - Shift - Graph - L

Lukt dit niet, dan is met behulp van ?CHR$ (201) uit te zoeken hoe deze kode wél te bereiken is. U zou toch al even kontroleren in hoeverre de aangegeven toetsenkombinatie voor uw computer de juiste is. Evenzeer kunt u in regel 3090 kodenummer 201 even overslaan door hiervoor een spatie te zetten.

Zodra alle regels zijn ingetikt, dan kan kort na de RUN de zaak weer worden afgebroken en regel 3090 worden geLIST. U ziet dan het ontbrekende deel linksboven van de witte pion op donkerblauw. Dan zal tenslotte door diverse toets-kombinaties te proberen dat deel verschijnen. Vergeet u dan alleen niet om de juiste kombinatie in APPENDIX-1 te noteren voor later gebruik. Een bepaalde karakter op het scherm te zetten mag geen hoofdbrekens kosten. Veel bordspelen kunnen op dezelfde wijze als schermbeeld gePRINT worden. Veel andere zaken ook uiteraard. U zult het nog gaan waarderen.

DATA-1 (regel 2940):
24-23-23-23-23-23-23-23
23-23-23-23-23-23-23-23
23-23-23-25-24-23-23-23
23-23-23-23-23-23-25-32

DATA-2 (regel 2950):
22- *32-97-32-98-32-99-32*

100-32-101-32-102-32-103-32
104-32-32-22-22-*82-79-67*
65-68-69-32-32-32-22-32

DATA-3 (regel 2960):
22-*56*-132-134-160-162-140-142
168-170-148-150-164-166-136-138
156-158-*56*-22-22-248-240-*32*
107-101-121-32-49-32-22-32

DATA-4 (regel 2970):
22-*32*-133-135-161-163-141-143
169-171-149-151-165-167-137-139
157-159-*32*-22-22-*82-69-77*
73-83-69-32-32-32-22-32

DATA-5 (regel 2980):
22-*55*-152-154-128-130-152-154
128-130-152-154-128-130-152-154
128-130-*55*-22-22-248-240-*32*
107-101-121-32-50-32-22-32

DATA-6 (regel 2990):
22-*32*-153-155-129-131-153-155
129-131-153-155-129-131-153-155
129-131-*32*-22-22-*79-80-71*
65-86-69-32-32-32-22-32

Voor de zwarte pion op lichtblauw zijn de kodes 128 t/m 131 gebruikt. In de voor sprites vereiste volgorde zijn deze kodes aldus als 128-130 in regel 2980 en als 129-131 in regel 2990 in de PRINTs gezet. Het geldt voor alle gebruikte kodes.

DATA-7 (regel 3000):
22-*54*-224-224-232-232-224-224
232-232-224-224-232-232-224-224
232-232-*54*-22-22-248-240-*32*
107-101-121-32-51-32-22-32

DATA-8 (regel 3010):
22-*32*-224-224-232-232-224-224
232-232-224-224-232-232-224-224
232-232-*32*-22-20-23-23-23
23-23-23-23-23-23-19-32

DATA-9 (regel 3020):
22-*53*-232-232-224-224-232-232
224-224-232-232-224-224-232-232
224-224-*53*-22-22-*91-32-32*
32-32-32-58-32-48-22-32

DATA-10 (regel 3030):
22-*32*-232-232-224-224-232-232
224-224-232-232-224-224-232-232
224-224-*32*-22-22-248-*32-32*
32-32-32-32-32-32-22-32

DATA-11 (regel 3040):
22-*52*-224-224-232-232-224-224
232-232-224-224-232-232-224-224
232-232-*52*-22-22-240-*32-32*
32-32-32-32-32-32-22-32

DATA-12 (regel 3050):
22-*32*-224-224-232-232-224-224
232-232-224-224-232-232-224-224
232-232-*32*-22-20-23-23-23
23-23-23-23-23-23-19-32

DATA-13 (regel 3060):
22-*51*-232-232-224-224-232-232
224-224-232-232-224-224-232-232
224-224-*51*-22-22-*32-84-69*
75-83-84-69-78-32-22-32

DATA-14 (regel 3070):
22-*32*-232-232-224-224-232-232
224-224-232-232-224-224-232-232
224-224-*32*-22-20-23-23-23
23-23-23-23-23-23-19-32

U zult in de voorgaande regel dat licht en donkerblauwe blokje herkend hebben, waaruit de niet-bezette vakjes van het speelveld bestaan. De kodes 224 en 232 zullen in het programma nog een belangrijke rol spelen.

DATA-15 (regel 3080):
22-*50*-176-178-200-202-176-178
200-202-176-178-200-202-176-178
200-202-*50*-22-22-*32-32-32*
32-32-32-32-32-32-22-32

DATA-16 (regel 3090):
22-*32*-177-179-201-203-177-179
201-203-177-179-201-203-177-179
201-203-*32*-22-22-*32-32-32*
32-32-32-32-32-32-22-32

DATA-17 (regel 3100):
22-*49*-204-206-184-186-212-214
192-194-220-222-188-190-208-210

180-182-*49*-22-22-*32-32-32*
32-32-32-32-32-32-22-32

DATA-18 (regel 3110):
22-*32*-205-207-185-187-213-215
193-195-221-223-189-191-209-211
181-183-*32*-22-22-*32-32-32*
32-32-32-32-32-32-22-32

DATA-19 (regel 3120):
22-*32-97-32-98-32-99-32*
100-32-101-32-102-32-103-32
104-32-32-22-20-23-23-23
23-23-23-23-23-23-19-32

DATA-20 (regel 3130):
26-18-23-23-23-23-23-23
23-23-23-23-23-23-23-23
23-23-18-27-22-*90-69-84*
91-32-32-32-32-32-22-32

DATA-21 (regel 3140):
32-22-*32-32-32-32-32-32*
32-32-32-32-32-32-32-32
32-32-22-*32*-22-248-*32-32*
32-32-32-32-32-32-22-32

DATA-22 (regel 3150):
32-22-*32-32-32-32-32-32*
32-32-32-32-32-32-32-32
32-32-22-*32*-22-240-*32-32*
32-32-32-32-32-32-22-32

DATA-23 (regel 3160):
32-26-23-23-23-23-23-23
23-23-23-23-23-23-23-23
23-23-27-*32*-26-23-23-23
23-23-23-23-23-23-27-32

Door de DATA-regels op de gebruikelijke manier in te tikken na de VPOKE-lus, passen we de direkte methode toe voor de realisering van het schermbeeld. Duidelijk mag zijn, dat de PRINT-regels aanzienlijk minder bytes vergen. Deze wat omslachtige manier om u de kodenummers aan te reiken, nodig voor de PRINT-regels, zal vooral vertrouwd maken met kodes en karakters. Want daarmee krijgt u veel te maken bij het VPOKEn.
Wanneer u er intussen in geslaagd bent alles in te tikken, dan kan de laatste aandacht hier regel 3170 gelden. De variabelen ZS (Zwart Score) en WS (Wit Score) duiken op. Een startwaarde hebben ze niet meegekregen. Het zal dan ook met nul beginnen en bij een volgend spel de

stand vasthouden. Dat geldt ook voor de tijd-printing, zoals in de INPUT waarden aan U1 en M1 zijn gegeven. Wel komen de klokwaarden voor zwart en wit in regel 90 weer op nul te staan, zoals ook ZET, maar de eenmaal opgegeven tijd blijft in U1 en M1 gehandhaafd. Deze PRINTs staan hier, omdat het deze routine is, welke bij het volgende spel opnieuw tot aktie wordt bewogen. Alles wat opnieuw gePRINT of gewist moet worden, dient onderaan zo'n routine te worden gezet. Soms hoeft niet de hele routine gebruikt te worden. Zolang het is afgesloten met een RETURN kan elke voorgaande regel via GOSUB worden bereikt. Zo kan er veelal aanzienlijk meer in een enkele routine worden ondergebracht.

Het is verstandig om een GOSUB naar de REM-regel te laten springen, Waarmee in elk geval een routine zal beginnen. Mocht er dan een regel toe te voegen zijn, dan kan dit altijd zonder het regelnummer van de GOSUB te moeten wijzigen. Soms denken we daar niet aan bij het invoegen of verwijderen van regels. Dan kloppen de regelnummers na GOSUB of GOTO niet meer en wordt het weer eens een kwestie van zoeken. Het gebruik van de REM-regel als regelnummer voor GOSUB en GOTO bespaart veel ergernis. En nu we het toch over invoegen en verwijderen hebben, is daar het verplaatsen mooi bij te betrekken. Ofwel we willen bepaalde regels ergens anders in het programma onderbrengen, óf we hebben dezelfde regels ook elders nodig. MSX ontslaat ons van de noodzaak om zulke regels opnieuw te typen. Het bestaande regelnummer overtikken met het gewenste zal zo'n regel op de goede plaats brengen. Na een LIST blijkt ook het eerste regelnummer er nog te zijn. En is die overbodig geworden, dan is die zoals bekend door het intypen ervan met RETURN te verwijderen. Het is erg handig en kan veel typwerk besparen. U wist het al? Dan praten we er niet meer over en typen de volgende regels in.

```
100 REM ** hoofdprogramma **
110 IF SP=1 THEN 120 ELSE 160
120 SZ=1:GOSUB 1100:ZET=ZET+1:LOCATE 25,
19:PRINT ZET
130 Z=6:KL=1:GOSUB 270
140 GOSUB 1140:SZ=2
150 GOSUB 270:GOTO 200
160 SZ=3:GOSUB 1120
170 Z=6:KL=15:GOSUB 270
180 SZ=4:GOSUB 1140
190 GOSUB 270
```

```
200 IF SP=1 THEN 210 ELSE 230
210 IF Z=5 THEN GOSUB 1820
220 GOTO 1850  ' schaak contr.
230 IF Z=5 THEN GOSUB 1830
240 GOTO 1850  ' schaak contr.
250 IF SP=1 THEN SP=0 ELSE IF SP=0 THEN
SP=1
260 GOTO 100
```

Geen gezoek naar de juiste toetsen meer. Dat schermbeeld behoort nu in de RAM-adressen te zitten. Vanaf deze plek gaan we ons met het eigenlijke programma bezig houden. Het hoofdprogramma is relatief kort en daarom overzichtelijk gebleven. Van hieruit wordt de routine opgeroepen, die aan het begin van de akties staat; GOSUB 270. Als het even kan, moeten we ons zo'n hoofdprogramma als het zenuwcentrum realiseren. Meestal kan dit ook. Daardoor komt er de nodige ordening in het programma. Van hieruit vertrekken we en hier keren we steeds weer terug. Er wordt vaak aan voorbij gegaan. Door het hele programma heen wordt er van de ene routine naar de andere gesprongen, die dan veelal ook nog in een weinig logische volgorde ergens zijn ondergeschoven. Probeer dan een programma maar eens te analyseren. Zoals de computer logische volgorde van bewerkingen eist, zo zullen we het ook van ons zelf verlangen bij de konstruktie van ons programma. In de praktijk maakt dit het programmeren alleen maar plezieriger en gaat het ons ook steeds vlotter af.

Wit heeft SP=1 gekregen en begint steeds, zoals te doen gebruikelijk is bij het schaken. Is het hoofdprogramma doorgelopen, dan vindt in regel 250 de wissel plaats. Sommigen zullen deze IF . . . THEN minder elegant vinden maar de goede werking ervan is belangrijker. Als wit aan zet is, dan krijgt deze de variabele SZ=1 of SZ=2 toegevoegd en u kunt raden, dat het voor Speler-Zet staat. Het gaat steeds om 2 akties. Eerst de aanwijzing van het te verplaatsen schaakstuk en daarna het zetten elders op het bord. Door ZS is zo te bepalen welke aktie er gaande is. En dat zal bij het doorlopen van routines hard nodig zijn. De zet wordt geteld en op het scherm weergegeven. Zo blijven we op de hoogte. Voordat naar de startroutine via GOSUB 270 wordt gesprongen, krijgt de variabele Z de waarde van 6 en voor hier wit KL de waarde 1. Die 6 is het nummer van onze aanwijssprite en KL de kleur zwart. Als zwart speelt, wordt die kleur wit (regel 170). Zo blijft het schaakstuk goed te onderscheiden en is het mede een aanwijzing wie aan zet is. Vooral de variabele Z is van belang. De waarden die het krijgt,

zijn steeds vlak en spritenummers. Het maakt alvast duidelijk, dat feitelijk die aanwijssprite zal veranderen in het schaakstuk dat we willen veranderen én zo ook over het bord kunnen verplaatsen. Maar die Z wordt ook gebruikt om een schaakstuk te identificeren. We zien dat in regel 210. Hier staat Z=5 voor de koning. In regel 90 zijn XP en PX gebruikt om de gebruikelijke positie van de koningen op het bord weer te geven in de overeenkomende schermlokaties. Wanneer nu een koning wordt verplaatst, dan is Z=5 aan de orde, dan dient de gewijzigde positie aan xp of px te worden gegeven. Immers het is altijd op deze positie, dat een koning kan worden gedreigd en een SCHAAK-situatie kan ontstaan. Zo dient de positie van de koning voortdurend te worden bijgehouden.
Alleen dan kan met sukses naar de schaakroutine worden gegaan via GOTO 1850. Zover zijn we hier nog niet. Er komen meer GOSUBS voor in ons hoofdprogramma. Ze verwijzen allen naar de tekst-routine, welke u hierna mag intypen. Omdat we wat krap in de schermruimte zitten, zijn we aangewezen op korte teksten. Die moeten bovendien op diverse regels gePRINT worden en veelal is daarvoor een al aanwezige tekst eerst te wissen. Daarom is het allemaal in één routine ondergebracht en konden alle teksten aan dezelfde variabele I$ worden toevertrouwd.

```
1080 REM ** teksten **
1090 I$="wit zet":RR=16:KK=22:GOTO 1330
1100 I$="wit zet":RR=16:KK=22:GOTO 1290
1110 I$="zwart zet":RR=16:KK=21:GOTO 133
0
1120 I$="zwart zet":RR=16:KK=21:GOTO 129
0
1130 I$="rocade!":KK=22:RR=17:GOTO 1310
1140 I$="welke zet":RR=17:KK=21:GOTO 133
0
1150 I$="gedaan!":KK=22:RR=16:GOTO 1310
1160 I$="remise!":KK=22:RR=15:GOTO 1290
1170 I$="geeft op!":KK=21:RR=15:GOTO 133
0
1180 I$="promotie!":KK=6:RR=21:GOTO 1290
1190 I$="[tijd om\":KK=21:RR=15:GOTO 133
0
1200 I$="nog eens?":KK=21:RR=16:GOTO 133
```

```
0
1210 I$="toets J/N":KK=21:RR=17:GOTO 133
0
1220 I$="[  WIT  \":KK=21:RR=14:GOTO 129
0
1230 I$="[  WIT  \":KK=6:RR=20:GOTO 1290
1240 I$="[ ZWART \":KK=6:RR=20:GOTO 1290
1250 I$="[ ZWART \":KK=21:RR=14:GOTO 129
0
1260 I$="WIT speelt":KK=5:RR=20:GOTO 133
0
1270 I$="zwart SPEELT":KK=4:RR=20:GOTO 1
330
1280 I$="verkeerde zet!":KK=3:RR=20:GOTO
 1330
1290 LOCATE 21,14:PRINT X$
1300 LOCATE 21,15:PRINT X$
1310 LOCATE 21,16:PRINT X$
1320 LOCATE 21,17:PRINT X$
1330 LOCATE KK,RR:PRINT I$
1340 FOR I=1 TO 10:BEEP:NEXT I
1350 RETURN
```

Voor de LOCATE-waarden moest RR en KK worden gebruikt, omdat de andere in aanmerking komende variabelen elders al worden gebezigd. Er mag natuurlijk geen verwarring ontstaan. Enige teksten zijn tweemaal nodig om bijvoorbeeld naar de juiste wisregels te kunnen springen. Die X$ gaven we in regel 20 SPACE$(9) mee. U hoeft niet vreemd op te kijken tegen die [en / in bijvoorbeeld regel 1230. Die karakters zijn in pijltjes veranderd.
De tekstroutine is een voorbeeld van een gekombineerde routine. Vooral de herhaald te PRINTen teksten zijn veel beter via een GOSUB te halen dan de tekst door het programma heen apart te vermelden met de LOCATE. De BEEP in de lus behoort de aandacht op een weergegeven tekst te vestigen. We kunnen het nog mooier doen door een tekst even aan en uit te PRINTen. Het wordt verderop gedaan. Voor het overige heeft deze tekstroutine weinig toelichting nodig. Hier en daar komt een uitroepteken achter een tekst voor. Wat in deze gevallen op z'n plaats is, kan echter al gauw hysterisch aandoen. Enige voorzichtig-

heid is geboden. Deze teksten zijn overigens niet de enige in het programma. Zoals herkend kan worden, was het nodig om die andere wél een eigen plek te geven bij de betreffende routines. Laten we de startroutine maar eens intikken.

```
270 REM ** aanwijzing **
280 F=STICK(0):REM of (1)/(2)
290 IF F=1 THEN Y=Y-2
300 IF F=2 THEN X=X+2:Y=Y-2
310 IF F=3 THEN X=X+2
320 IF F=4 THEN X=X+2:Y=Y+2
330 IF F=5 THEN Y=Y+2
340 IF F=6 THEN X=X-2:Y=Y+2
350 IF F=7 THEN X=X-2
360 IF F=8 THEN X=X-2:Y=Y-2
370 PUT SPRITEZ,(X,Y),KL,Z
380 KEY(1)ON:KEY(2)ON:KEY(3)ON
390 ON KEY GOSUB 1590,1690,1730
400 GOSUB 2450 ' klok
410 IF STRIG(0) THEN GOSUB 430 ELSE 280:
REM of (1)/(2)
420 RETURN
```

Het typt in elk geval prettiger, als we dat steeds met plukjes regels kunnen doen. Deze routine zal u niet onbekend zijn. Daarmee verplaatsen we een sprite over het scherm. De REM in regel 280 en 410 maakt duidelijk, wat u te doen hebt, om inplaats van cursortoetsen en spatiebalk de joystick en vuurknop te gebruiken bij het schaken. Er kan dus ook diagonaal mee verplaatst worden. Eigenlijk is het een nogal omslachtige routine. Het zou in één enkele regel samengevat moeten kunnen worden. Wellicht ontdekt u die regel binnenkort. In regel 370 komt de aanwijssprite opdagen, die gedurende de loop van het spel vele malen van gedaante kan wisselen. In het hoofdprogramma werden aan Z en KL al waarden toegekend. En in regel 90 ook de startwaarde van X en Y. Ook deze routine is kort en krachtig. Het is terecht startroutine te noemen, omdat we zelf akties op gang brengen door of spatiebalk/vuurknop óf de 3 funktietoetsen in te drukken. En zolang we dat niet doen, zal de aanwijssprite te verplaatsen zijn naar de gewenste plaats op het schaakbord. Zo is de routine het voorportaal van de verdere akties, die het ons tot een plezierige bezigheid moeten ma-

ken een partij schaak te spelen. Meer moet er niet in zo'n routine worden gezet.
De ON KEY instruktie is compacter dan INKEY$ en hier goed te gebruiken. Ofschoon aan schermlezing wordt gedaan in het programma, is voor 3 funktietoetsen gekozen, omdat hiervan slechts een enkele maal gebruik behoeft te worden gemaakt. MSX-Basic biedt veelal alternatieve mogelijkheden en terwille van wat ervaring is het goed, om eens van andere instrukties gebruik te maken. Volgen we eerst eens de GOSUBS van de KEY's.

```
1590 REM ** rocade **
1600 GOSUB 1130
1610 IF SP=1 THEN 1650 ELSE 1620
1620 E=E+1:IF E=2 THEN 1640
1630 SP=1:SZ=1:GOSUB 1090:RETURN
1640 GOSUB 1150:FOR D=1 TO 75:BEEP:NEXT
D:GOSUB 1100:RETURN
1650 F=F+1:IF F=2 THEN 1670
1660 SP=0:SZ=3::GOSUB 1110:RETURN
1670 GOSUB 1150:FOR D=1 TO 75:BEEP:NEXT
D:GOSUB 1120:RETURN
1680 REM ** remise **
1690 GOSUB 1160
1700 WS=WS+.5:LOCATE 22,21:PRINT WS:ZS=Z
S+.5:LOCATE 22,20:PRINT ZS
1710 GOTO 1760
1720 REM ** opgave/mat **
1730 IF SP=0 THEN 1740 ELSE 1750
1740 GOSUB 1250:GOSUB 1170:WS=WS+1:LOCAT
E 22,21:PRINTWS:GOTO 1760
1750 GOSUB 1220:GOSUB 1170:ZS=ZS+1:LOCAT
E 22,20:PRINT ZS
1760 GOSUB 1200:GOSUB 1210
1770 A$=INKEY$:IF A$="" THEN 1770
1780 IF A$=CHR$(74) THEN 70
1790 IF A$=CHR$(78) THEN 3190
1800 GOTO 1770
```

Omdat wit en zwart automatisch afwisselend aan zet komen, is het aldus voor een rokade nodig om de speler in kwestie een extra zet te geven. Vast te stellen is, wie de rokade geldt en of al eerder een rokade plaats vond. Voor het laatste is simpelweg teller E en F gebruikt. Staan die op 2, dan gaat het feest niet door en komt alleen de tekst "gedaan!" op het scherm. Omdat de variabele F ook gebruikt wordt in de joystick-routine is een absolute bepaling hier beter dan IF F< 1.Voor de rest is het simpel. Wanneer wit aan zet is, terwijl zwart na zijn torenzet een rokade wil, dan krijgt zwart opnieuw de variabelen SP=0 en SZ=3 toegewezen en wordt naar het hoofdprogramma geRETURNed. Zo kan zwart de rokade uitvoeren en voorkomt teller F een herhaling. Strikt genomen zou zwart het toch nog eens kunnen proberen. Immers, als F <2 is dan geldt de voorwaarde niet meer. Hier is ervan uitgegaan, dat die ene waarschuwing "gedaan" zo'n poging tot oplichting zal voorkomen. Wilt u het helemaal veilig stellen, dan moet voor F een andere variabele worden gekozen en is aldus <1 te gebruiken.
Zaak is alleen bij telvariabelen, dat deze niet ergens anders voor worden gebruikt en dat ze waar nodig steeds weer op 0 worden gezet.

De remiseroutine is even simpel. In regel 1690 wordt hiervoor weer uit de tekstroutine geput. Het belangrijkste is, dat wit zowel als zwart dat halve punt bij hun score krijgen geteld. Tenslotte wordt verwezen naar de INKEY$-routine, welke gedeeld wordt met de routine van de 3e KEY voor opgave/schaakmat. Ook hier weer het noodzakelijke onderscheid tussen wit en zwart, dat dankzij SP=0 en SP=1 mogelijk wordt. Wie aan zet is en zich gewonnen geeft, staat daarmee een punt aan de ander af. Hier zowel als bij een remise betekent dit tegelijk het einde van de partij. De INKEY$ instruktie laat kiezen tussen een nieuwe partij of beëindiging van het programma. In het eerste geval wordt opnieuw gestart met regel 70. Hoewel het hele scherm feitelijk opnieuw gePRINT wordt, voltrekt zich dit dusdanig snel, dat het de indruk wekt, alsof alle schaakstukken weer op hun startposities worden teruggezet. Ook de aanwijssprite neemt de startpositie weer in. ZET en de klok van zwart en wit komen weer op 0. Uiteraard blijft de score wél gehandhaafd. Zoals ook de eenmaal opgegeven speeltijd. Op deze wijze is heel snel een volgende partij te beginnen. Het programma behoeft er niet voor te worden afgebroken en opnieuw met RUN in werking te worden gezet. Als het maar enigzins mogelijk is, moeten we dit proberen in te bouwen. Het is erg vervelend, wanneer bij ook nog een relatief kort programma alles opnieuw geRUNd moet worden voor een volgend gebruik. Het zal in de meeste gevallen geen noodzaak zijn. Het definitieve einde van een programma behoort door de gebruiker bepaald te kunnen worden. Dat gebeurt hier, als in regel 1790 de N voor "nee" wordt ingedrukt. Weer een paar regels typwerk voor u:

```
3190 REM ** einde spel **
3200 SCREEN0:CLS:COLOR 15,4,4
3210 DEFUSR=&H69:A=USR(0)
3220 LOCATE 10,12:PRINT"tot ziens!"
3230 FOR D=1 TO 500:NEXT D
3240 CLS:KEY ON:END
```

Het zijn de slotregels van het programma. Natuurlijk kan zo'n routine overal in het programma staan. De meest logische plaats zal evenwel tegelijk de afsluiting van een programma zijn. END is END en het is wat merkwaardig, als we dan tóch nog een serie programmaregels laten volgen. Zo'n END-routine kunnen we al in het begin van de programma-opbouw schrijven. Mét de bijbehorende INKEY$-toets. Proberen we dan stukjes programma, die nog verbetering behoeven, dan krijgen we met een drukker op die toets weer een schoon standaardscherm. Eventuele sprites verdwijnen en van vorm gewijzigde karakters komen weer in hun originele gestalte tevoorschijn. We zullen algemeen de gebruiker van ons programma niet laten zitten met een rommelig scherm. Het hoort bij de goede verzorging, om het scherm weer netjes op te leveren, alsof er geen programma in aktie is geweest. De gebruiker behoeft dan alleen nog NEW in te tikken. Uiteraard kunnen we dit NEW ook in zo'n END-routine zetten. Daarmee gaan we echter op de stoel van de gebruiker zitten. Wellicht wil die het programma nog even listen of er iets aan veranderen. Maar niet doen dus. Het is trouwens toch al moeilijk, om in MSX onderscheid te maken tussen kommando's en funkties. Alles kan in een programma worden opgenomen, al is het niet altijd even zinvol. Daarom heb ik het steeds over instrukties als meer algemene term. Iets anders is het gebruik van NEW in bijvoorbeeld een programma met machinekode . We hebben er doorgaans een kort hulpprogramma in Basic voor nodig, dat na het POKEn van de machinebytes geen funktie meer heeft. Sluiten we het dan af met NEW, dan komen we er netjes vanaf.

Een hulpmiddel bij het programmeren kan TRON zijn. Een groot voordeel biedt MSX hier doordat deze instruktie in het programma kan worden opgenomen aan het begin van de regels, waarvan we de verwerkingsvolgorde willen kontroleren. Dat is lang niet altijd het geval. In MSX is zo'n plaatselijk gebruik mogelijk. Als het wat ingewikkeld geworden is met veel verwijzingen, voorwaarderegels en variabelen, dan kan TRON een handig hulpmiddel zijn. Het brengt de regelnummers op het scherm in de volgorde van lezing en verwerking. Door naderhand TROFF in te tikken schakelen we dat weer uit. Toch zal TRON niet al te vaak worden gebruikt. De fouten die we kunnen

maken, zitten veelal niet in een verkeerde verwerkingsvolgorde. In de meeste gevallen gaat er iets mis met gebruikte variabelen en/of daaraan toegekende waarden. Een computer dwingt tot logisch denken. Dat is er tegelijk het meest positieve aspekt van. Het houdt onze hersenen in konditie. Voegen we daarbij dan nog het kreatieve aspekt en de vindingrijkheid, dan hebben we speciaal aan onze MSX-computer een schier onuitputtelijk instrumentarium.
Het maakt programmeren telkens weer tot een boeiende uitdaging. Hier is dit schaakbordprogramma niets meer dan een voorbeeld, waaraan u overigens óók een aardig tijdverdrijf kunt overhouden. Geen twee programma's zijn gelijk. Ze hebben alleen veel Basic-instrukties gemeenschappelijk en meestal ook hun manier van opbouw. Andere variabelen met weer andere waarden doen niets af aan dit voorbeeld. Er is in elk programma even zorgvuldig mee om te gaan. Veel doen met weinig zal het motto blijven. Dat een programma goed draait, behoeft nog niet te betekenen, dat het ook goed is opgezet. We moeten niet al te snel tevreden zijn. Een streven naar perfektie resulteert op de duur in een steeds betere konstruktie en handiger gebruik van Basic-instrukties. Dat kost tijd en aldus ervaring. Maar een meer bescheiden opstelling moeten we ons aanvankelijk beslist eigen maken. De oefening baart ook hier kunst.
Aarzelt u dus niet, wanneer u in dit programma verbeteringen meent te kunnen aanbrengen. Daar is het voor. Zoals het ook de mogelijkheid zal bieden tot uitbreiding. Ja, zelfs zodanig dat u ermee kunt schaken tegen de computer. Het ligt er maar aan hoe goed u bent in het formuleren van daarvoor benodigde regels.

In regel 400 van de startroutine wordt naar de klokroutine verwezen. Dat gebeurt telkens weer. Totdat we via spatiebalk/vuurknop of één der funktietoetsen de routine verlaten. Wilt u deze regels even toevoegen aan wat u tot nu toe al in de RAM-adressen hebt getypt?

```
2450 REM ** klok **
2460 IF SP=1 THEN 2470 ELSE 2550
2470 X1=X1+1:IF X1=7 THEN S1=S1+1:X1=0
2480 IF S1=60 THEN M2=M2+1:S1=0
2490 IF M2=60 THEN U2=U2+1:M2=0
2500 LOCATE 22,10:PRINT USING"##:##:##";
U2,M2,S1
2510 IF U2=U1 AND M2=M1 THEN 2520 ELSE 2
600
2520 IF SP=1 THEN 2530 ELSE 2540
```

```
2530 GOSUB 1220:GOSUB 1190:ZS=ZS+1:LOCAT
E 22,20:PRINT ZS:GOTO 1760
2540 GOSUB 1250:GOSUB 1190:WS=WS+1:LOCAT
E 22,21:PRINT WS:GOTO 1760
2550 X2=X2+1:IF X2=7 THEN S2=S2+1:X2=0
2560 IF S2=60 THEN M3=M3+1:S2=0
2570 IF M3=60 THEN U3=U3+1:M3=0
2580 LOCATE 22,9:PRINT USING"##:##:##";U
3,M3,S2
2590 IF U2=U1 AND M3=M1 THEN 2520
2600 RETURN
```

We hebben te maken met twee klokken. De variabele SP zorgt voor het onderscheid tussen zwart en wit. Doet een variabele dit ook eens een keer. We komen hier dus die variabelen tegen, die in regel 90 hun startwaarde kregen; allemaal op 0. Nu is het lastig, om een klok enigzins volgens de werkelijkheid te laten lopen. Wanneer we uit de startroutine stappen, dan funktioneert de klok even niet meer. Om gemiddeld toch redelijk in de buurt te blijven bij de werkelijke tijdmeting, is hier van de variabele X1 of X2 gebruik gemaakt. Het vertraagt de tijdmeting, net genoeg om bijvoorbeeld een uur speeltijd ook werkelijk 60 minuten te laten duren. Alleen als u meer routines zou inbouwen, dan moet deze vertraging worden aangepast en wellicht geheel verdwijnen. Het is met behulp van een horloge te kontroleren. De regels 2470 t/m 2490 geven er een voorbeeld van, dat na een IF . . . THEN uitsluitend kan volgen, wat van toepassing is indien aan de voorwaarde wordt voldaan. Zolang daarvan geen sprake is, wordt er eenvoudig niet verder gelezen op zo'n regel. De PRINT USING instruktie is hier gebruikt, om de weer te geven tijd op de gewenste plek gePRINT te krijgen.
Doen we dit niet, dan zal aan weerszijden van een weer te geven getal ruimte gereserveerd worden voor zowel een groter getal als een negatieve waarde middels het minteken. Nu bepalen we een vaste plaats. Wanneer we krap in de schermruimte zitten of PRINTen op een kleurvlak, dan voorkomt PRINT USING een ongewenste uitloop en/of een ongewenst wissen.

De tijd wordt op het scherm ook in sekonden weergegeven. Zo blijft het aktief op het scherm. De uurtijd en minutentijd wordt steeds vergeleken met de bij de INPUT opgegeven speeltijd in U1 en M1. Zodra dit aan elkaar gelijk is, komen er weer tekstjes op het scherm en wordt opnieuw gebruik gemaakt van de INKEY$-routine. Wat we toch

al aan routines hebben, dan zullen we ook zo dikwijls als nodig is gebruiken. Zou wit de speeltijd als eerste overschrijden, dan levert dit zwart een punt op. Het is naast remise en opgave/mat de derde manier waarop een partij kan eindigen. Daarom ook de uitnodiging om de J-toets in te drukken voor nog een partij of de N-toets om er een punt achter te zetten. Zo'n tijdmeting is zowel goed voor een partij snelschaak als om ons te binden aan een tijdlimiet. Spelen we schaak met een partner, die nogal veel bedenktijd nodig heeft, dan dwingen we zo tot wat snellere beslissingen. Spelen tegen de klok maakt het spel des te boeiender.

Opgemerkt kan nog worden, dat zowel in de startroutine als in de routine die u hierna dient in te tikken, achter belangrijke GOSUBS voorbij de apostrof de betekenis daarvan is vermeld. Al dwingt dit om voor elke GOSUB een aparte regel te schrijven, het maakt voor ons én voor hen die ons programma wellicht ooit zullen analyseren de zaken aanzienlijk eenvoudiger te herkennen. Zo weten we altijd, om welke routine het gaat en hoeven we die niet speciaal op te zoeken. Het "wat is dat toch voor een routine" beantwoordt zich op deze manier zelf. Zolang we nog royaal in de vrije bytes zitten, hebben we geen reden om ons zo'n hulpmiddel te ontzeggen. En al marcheert het allemaal keurig in de pas met ons programma, laat zo'n aanwijzing toch maar staan. Dan bewijst het ons later dienst bij het doorlopen van een "bejaarde" listing.

Het ligt voor de hand, dat we in een speelbordprogramma aan schermlezing gaan doen. Het "wat staat waar" laat zich middels VPEEKen van de schermlokaties in BASE (5) nu eenmaal verreweg het eenvoudigste en het snelste vaststellen. Daarvoor zouden we anders tientållen IF . . .THEN regels nodig hebben. Zo is deze schermleesroutine het hart van een programma te noemen. Dat is het des te meer, omdat we hier niet via de toetsen ingeven vanaf welke positie we welke zet willen plegen.
Verreweg de meeste van dergelijke programma's en soms ook professionele werken met het toetsenbord. Op die manier kunnen ingetypte waarden meteen worden verwerkt. Een bordprogramma heeft het evenwel in zich om van de speler enig nadenken te eisen. Dat doen we gemakkelijker in de luie stoel dan bovenop de computer. En tenslotte hebben we in de joystick onze afstandbediening bij de hand. Het zou veelvuldiger worden toegepast, indien er voldoende bekendheid met die schermlezing zou bestaan. En dat is vaak niet het geval. Ofschoon de routine natuurlijk hier speciaal ons schaakbord betreft, is het principe ervan identiek voor elk bordspel.

```
430 REM ** schermlezing **
440 K=INT(X/8):R=INT(Y/8)+1
450 A=BASE(5):C=VPEEK(A+(R*32+K))
460 IF SZ=2 OR SZ=4 THEN 600
470 GOSUB 1360:GOSUB 830
480 REM ** verwerking **
490 IF C<152 THEN W=1:GOTO 530
500 IF C<176 THEN W=2:GOTO 530
510 IF C<200 THEN W=1:GOTO 530
520 IF C<224 THEN W=2
530 N=(C-124)/4:C2=C
540 GOSUB 2210   ' zetcontrole
550 ON N GOSUB 930,940,960,950,970,980,9
30,940,960,950,970,980,930,940,960,950,9
70,980,930,940,960,950,970,980
560 GOSUB 1430   ' sprite weg
570 GOSUB 1390   ' sprite terug
580 ON W GOSUB 1000,1020
590 GOTO 820
600 IF C<152 THEN W=1:GOTO 650
610 IF C<176 THEN W=2:GOTO 650
620 IF C<200 THEN W=1:GOTO 650
630 IF C<224 THEN W=2:GOTO 650
640 IF C=>224 THEN 660
650 ON W GOSUB 1000,1020
660 A=BASE(5):C=VPEEK(A+(R*32+K))
670 IF C2<152 THEN CC=1:GOTO 710
680 IF C2<176 THEN CC=2:GOTO 710
690 IF C2<200 THEN CC=1:GOTO 710
700 IF C2<224 THEN CC=2
710 ON CC GOSUB 730,750
720 GOTO 770
730 IF C=232 THEN C=C2+24 ELSE C=C2
740 RETURN
750 IF C=224 THEN C=C2-24 ELSE C=C2
760 RETURN
770 GOSUB 830    ' registratie
```

```
780 GOSUB 990    ' slot zettekst
790 GOSUB 1040   ' stuk printen
800 GOSUB 1430   ' sprite weg
810 GOSUB 1450   ' promot. contr.
820 RETURN
```

Wellicht is u bij het intypen opgevallen, dat er hier wat ontbreekt. Het behoort u althans op te vallen. Tussen regel 440 en 450 behoort eigenlijk een voorwaarde te staan, welke een aanwijzing buiten het bord voorkomt.
Nu kunnen we de aanwijssprite ook buiten het bord zetten, waardoor het allemaal lelijk de mist in kan gaan. Mocht u wat minder vertrouwen hebben in een korrekte speelwijze, dan is daartoe eenvoudig te dwingen.

```
IF K<1 OR K>17 OR R<1 OR R>17 THEN 820
```

En regel 820 bevat de RETURN. U kunt er desgewenst een tekstje aan verbinden zoals "doe gewoon, druiloor" met een waarschuwend deuntje. Het mag een voorbeeld zijn van de knutselmogelijkheden in dit programma.

Bij deze routine moeten we toch wel even langer stilstaan. Omdat u ongetwijfeld in daartoe geschikte programma's een dankbaar gebruik van de schermlezing zult maken. En daarmee is veel mogelijk. De startregels 440 en 450 kunnen u bekend voorkomen. De positie van de aanwijssprite wordt via INT door 8 gedeeld om de juiste R en K waarde te verkrijgen. In dit geval moest er bij R nog 1 worden bijgeteld. Het is snel uit te vinden, als we even van een hulpregel gebruik maken.

```
LOCATE 0,22:PRINT "K=";"R=";K;R
```

Wat er dan op het scherm wordt gezet, moet kloppen met de exacte kolom en regelpositie op het schaakbord. Hetzelfde kan ook gedaan worden met de C-waarde in VPEEK van de volgende regel. Op de plek waar we de aanwijssprite neerzetten, moet die C-waarde overeenstemmen met het gebruikte kodenummer op die plaats. Voor een leeg lichtblauw vakje behoort dat aldus kode 224 te zijn. Vanzelfsprekend overtuigen we ons er uitvoerig van, dat de zo aan te wijzen lokaties en kodes in overeenstemming met onze goede bedoelingen zijn. Pas als dit korrekt is, kunnen we ons gaan bezighouden met de verdere opbouw van de routine. Dán is zo'n hulp PRINT-regel snel genoeg weer te verwijderen. Zeker, bolleboosjes berekenen een en ander op papier. Hoe bolleboos we echter ook mogen zijn, de computer zal ons hoe

dan ook de baas zijn. Waarom die dan niet het werk laten doen?

Gelezen wordt alleen de schermlokatie waarop de aanwijssprite zich bevindt. Daarom voltrekt het zich zeer snel. Het doorlopen van de routine is een kwestie van minder dan een sekonde. Dat moet ook, want anders raakt onze klok van slag. We komen in regel 460 de variabele SZ tegen, die het mogelijk maakt een deel van de routine te passeren. In SZ=2 en SZ=4 is te herkennen, dat wit of zwart daarmee aan de tweede aktie bezig zijn. Het zetten op een nieuwe positie. Is dat niet het geval, dan wordt het eerste onderdeel van de routine wél verwerkt. Voordat we ons daarin gaan verdiepen, moeten we even dat kladje raadplegen, waarop we hebben genoteerd welke kodes we voor onder andere welk schaakstuk hebben gekozen. Dat kunnen en hoeven we niet allemaal te onthouden. Ook de aan de variabele Z toe te kennen waarde wil genoteerd zijn voor ons programmeergemak.
Dat kladje hebben we verworven, toen we middels het ontwerpprogramma PRENT de DATA-regels noteerden. Het hoort er zo uit te zien:

	pion	toren	paard	loper	dame	koning
z/1. bl.:	128	132	136	140	144	148
z/d. bl.:	152	156	160	164	168	172
w/l. bl.:	176	180	184	188	192	196
w/d. bl.:	200	204	208	212	216	220

Z=0 (pion) –Z=1 (toren) –Z=2 (paard) –Z=3 (loper) –Z=4 (dame) –Z=5 (koning) –Z=6 (aanwijssprite)
kode 224 voor lichtblauw vakje / kode 232 voor donkerblauw vakje.

Met deze kodenummers moeten we werken. En dat begint meteen al in de regels 490 t/m 530. De W staat hier voor Wis en wissen moeten we. Het kladje maakt duidelijk, dat kodenummers lager dan 152 lichtblauw als vakkleur hebben en we geven W de waarde 1. Het wordt 2 voor de kodes, die het donkerblauwe adoreren. Met dat duo springen we in de regel 580 naar de routine, waarmee het vakje van het te verplaatsen schaakstuk in de gewenste blauwtint wordt gezet. Daarmee verdwijnt dat stuk van het bord om elders opnieuw gePRINT te worden. Laten we die paar regels van de wisroutine maar even intypen. Dan is dat ook maar weer gebeurd.

```
1000 REM ** vakje wissen **
1010 LOCATE K,R:PRINT"αα":LOCATEK,R+1:PR
INT"αα":GOTO 1030
```

```
1020 LOCATE K,R:PRINT"■■":LOCATE K,R+1:P
RINT"■■"
1030 RETURN
```

Amper de type-moeite waard en desondanks onmisbaar in het programma. Laten we evenwel niet te rap van stapel lopen. In regel 470 verwijst GOSUB 1360 ook naar een wisroutine van nog bescheidener formaat. Deze is er om tekst in het kader onder het schaakbord te wissen. Op die plaats dient gePRINT te worden welk stuk we van welke positie naar welke positie verplaatsen. En als u dat wilt én een printer bezit, dan kunt u in de betreffende routines de regel toevoegen, die deze zetregistratie ook op papier gaat afdrukken.
Toch maar eerst even deze 3 regels intikken a.u.b..

```
1360 REM ** wis-routine **
1370 Y$=SPACE$(16):LOCATE 2,20:PRINT Y$:
LOCATE 2,21:PRINT Y$
1380 RETURN
```

Alweer gedaan. De tweede GOSUB in regel 470 verwijst naar een routine, die wat meer typwerk gaat vragen. We geven een zet nu eenmaal aan als bijvoorbeeld a4 of g6. En dankzij de afstandsbediening hebben we hier wél de nodige IF . . . THEN regels nodig.

```
830 REM ** registratie **
840 IF R=2 THEN L=8 ELSE IF R=4 THEN L=7
850 IF R=6 THEN L=6 ELSE IF R=8 THEN L=5
860 IF R=10 THEN L=4 ELSE IF R=12 THEN L
=3
870 IF R=14 THEN L=2 ELSE IF R=16 THEN L
=1
880 IF K=2 THEN K$="a" ELSE IF K=4 THEN
K$="b"
890 IF K=6 THEN K$="c" ELSE IF K=8 THEN
K$="d"
900 IF K=10 THEN K$="e" ELSE IF K=12 THE
N K$="f"
910 IF K=14 THEN K$="g" ELSE IF K=16 THE
N K$="h"
920 RETURN
```

Deze routine moet terwille van de zetregistratie telkens weer worden gelezen. Ook bij de tweede aktie van het zetten, zoals in regel 770 is vermeld. In dit geval is niet aan IF . . . THEN te ontkomen. De zo vast te stellen waarden voor L en K$ worden aktueel in de regels 530 t/m 550. Weer om die voorwaarderegels te vermijden, wordt aan N een waarde gegeven variërend van 1 t/m 24. Veel beter nog zou het zijn geweest, indien voor de 4 kodes die kunnen staan voor elk schaakstuk slechts één getal had kunnen worden verkregen. Dán zouden er in regel 550 niet 24 doch slechts 6 GOSUBS mogelijk zijn geweest. Nu moet de 6 viermaal in de regel zijn. Misschien lukt het u wél om daarvoor een andere formulering achter N te vinden.
Laten we in ieder geval deze routines maar eens intypen.

```
930 LOCATE 2,20:PRINT"PION van ";K$;L:LO
CATE 2,21:PRINT"naar ":Z=0:RETURN
940 LOCATE 2,20:PRINT"TOREN van ";K$;L:L
OCATE 2,21:PRINT"naar ":Z=1:RETURN
950 LOCATE 2,20:PRINT"LOPER van ";K$;L:L
OCATE 2,21:PRINT"naar ":Z=3:RETURN
960 LOCATE 2,20:PRINT"PAARD van ";K$;L:L
OCATE 2,21:PRINT"naar ":Z=2:RETURN
970 LOCATE 2,20:PRINT"DAME van ";K$;L:LO
CATE 2,21:PRINT"naar ":Z=4:RETURN
980 LOCATE 2,20:PRINT"KONING van ";K$;L:
LOCATE 2,21:PRINT"naar ":Z=5:RETURN
990 LOCATE 7,21:PRINT K$;L:RETURN
```

Daarmee zal de zet worden gePRINT. En in deze regels kunt u een printer-instruktie toevoegen. Hier ook krijgt Z de bij een stuk behorende waarde toegekend, zoals we deze voor onder andere de sprite moeten gebruiken. Eigenlijk hoort regel 990 er nog niet bij. Daarmee komt de slottekst van de zetregistratie op het scherm. Dat wordt in regel 780 pas aktueel. Keren we nog even terug naar regel 530. Aan C2 wordt de C-waarde toegekend, zoals via VPEEK opgespoord werd. Omdat die waarde bij de volgende lezing zal veranderen, moeten we deze vasthouden in C2. Daarin zit immers de kode van het te verplaatsen en naderhand te herPRINTen schaakstuk. Die mogen we in geen geval kwijt raken.
Dan is er in regel 540 de GOSUB voor de zetkontrole. Ook meteen maar intikken. Het zijn onrustige pagina's.

```
2210 REM ** zet controle **
2220 IF SP=1 AND C<176 THEN 2240 ELSE IF
 SP=0 AND C>172 THEN 2250
2230 RETURN
2240 GOSUB 1360:GOSUB 1260:SP=1:GOTO 110
2250 GOSUB 1360:GOSUB 1270:SP=0:GOTO 110
2260 REM ** verkeerde zet **
2270 REM
2280 REM
```

Ja, wat al te simpel zo. Als wit (SP=1) speelt, terwijl de C-waarde bij een zwart stuk hoort, dan zit wit zwaar fout. Teksten attenderen daarop en het behoort zwart te zijn die gaat zetten. Of andersom natuurlijk. Meer doet de routine niet. Maar de REM-regels nodigen u uit, om daarin verandering te brengen. De routine is uit te breiden door de zetmogelijkheden van elk schaakstuk te formuleren. Wordt hieraan dan niet voldaan, dan zou dit kunnen resulteren in bijvoorbeeld "gij speelt vals" en "nog maar eens en nu volgens de regels". Omdat een beetje ervaren schaker zoiets niet mag overkomen, is de routine hier beperkt. Echter vooral beginners zouden dank zij zo'n uitbreiding wél de zetmogelijkheden leren kennen. Het is de moeite waard om die aanvulling te overwegen. Wacht er dan nog eventjes mee. Want wellicht komt u verderop nog op glansrijke ideetjes.

We zitten nog steeds in de schermleesroutine en nu bij de regels 560 en 570. Sprite weg en sprite terug? Het zou merkwaardig zijn, indien het niet om dezelfde sprite ging. Goed voor een paar regels typwerk.

```
1390 REM ** sprite routine **
1400 IF SP=1 THEN KL=15 ELSE IF SP=0 THE
N KL=1
1410 PUT SPRITE 6,(X,209),0,6
1420 PUT SPRITE Z,(X,Y),KL,Z:GOTO 1440
1430 PUT SPRITE Z,(X,209),0,Z
1440 RETURN
```

De sprite die in regel 1430 weg moet via de Y-waarde 209 is onze aanwijssprite. Dat wordt nog eens gedaan in regel 1410 met het onmiskenbare vlaknummer 6. Daarmee laat zich de toestand zo weergeven. Het schaakstuk is gewist en het vakje is helemaal in de blauwe tint gezet. De aanwijssprite is zolang in de VUT gedaan. Maar de tweede GOSUB

haalt zo vlot de nieuwe sprite terug, dat we van dit dramatische gebeuren amper iets bemerken. Want het betreffende schaakstuk heeft een Z-waarde gekregen en dat kennen we als vlaknummer aan die sprite toe. En ziedaar ons schaakstuk is er weer. In de hoedanigheid van sprite. Nu, zo'n apparaat is wél te bewegen over het schaakbord en dat zal dan ook worden gedaan in de tweede aktie.
Zo maken we van de nood een deugd. Al kunnen 8 schaakstukken slechts naast elkaar gePRINT worden, bewegen kunnen we ze lekker toch dankzij die ene sprite. Dat maakt het heel wat plezieriger, dan alleen wissen en herprinten van een stuk. U kunt in de spriteroutine andermaal konstateren van hoeveel nut ons de variabele SP is. Hier bepalen we er de kleur van de sprite mee. De eerste aktie is afgesloten en we zijn met het als sprite getransformeerde schaakstuk terug in de startroutine. Een tekstje vraagt, welke zet we voornemens zijn te plegen. Wie dan de schaakstuksprite netjes op het gewenste vakje heeft gezet en spatiebalk/vuurknop heeft bewerkt, die heeft daarmee het licht op groen gezet voor de tweede aktie.

Voor wit is dan SZ=2 en voor zwart SZ=4 aan de beurt gekomen. Dat voert ons naar het tweede deel van onze schermleesroutine. Dat zat er dik in. We hebben in regel 450 een nieuwe C-waarde ontvangen. Nu van het vakje waarin we willen zetten. Dat kan een leeg vakje zijn. Of bezet door een stuk van de vijand. Daarom kennen we andermaal de W-waarde toe aan de sets kodes en gaan eerst eens zo'n vijandig stuk om zeep brengen. Is het onbezet, dan schieten we door naar regel 660 voor een nieuwe schermlezing. Daarvan staat bij voorbaat vast, dat alleen kode 224 voor het lichte of kode 232 voor het donkere vakje geregistreerd zal worden. Want al stond er een schaakstuk, daarvan is nu alleen de achtergrondkleur als vakkleur overgebleven. Eerst spelen we op nummer zeker door aan de oorspronkelijke, aan C2 toevertrouwde kode de variabele CC toe te kennen voor eveneens het wissen. WW had ook gekund, maar er zitten toch al zoveel mensen zonder werk in ons land. Zo kleuren we het gekozen zetvakje nog eens op de vereiste manier in. Zodra dat is gebeurd, wandelen we langs de GOSUBS vanaf regel 770 en daarvan hebben we GOSUB 830 en 990 al gehad. Daarom eventjes terug naar die CC-GOSUBS. Het zou tamelijk overbodig zijn, om zo'n zetvakje herhaald te kleuren. We willen daarin het bewuste schaakstuk PRINTen. Als we nu het kladje met de kodenummers bekijken, dan blijkt bijvoorbeeld het nummer van de zwarte pion op donkerblauw 24 hoger te zijn dan dat van dezelfde meneer op lichtblauw. Dat geldt voor alle kodenummers van diverse schaakstukken. Daarmee valt wat te doen. Wanneer VPEEK als C-waarde 232 heeft geregistreerd en de oorspronkelijke aan C2 toegekende C-waarde was bijvoorbeeld 128, dan is het daarbij gegaan om een stuk met een licht-

blauwe achtergrond. De nieuwe C-waarde 232 verlangt echter een donkerblauwe achtergrond. Geen probleem. We maken van de nieuwe C-waarde simpelweg de waarde van C2 plus 24. Want dan klopt de kleur. En andersom geldt dit natuurlijk ook, zoals in regel 750 is samengevat. Alweer heel wat IF . . . THEN regels bespaard.

```
1040 REM ** stuk printen **
1050 A$=CHR$(C):B$=CHR$(C+1):C$=CHR$(C+2
):D$=CHR$(C+3):BEEP
1060 LOCATE K,R:PRINT A$:LOCATE K,R+1:PR
INT B$:LOCATE K+1,R:PRINT C$:LOCATE K+1,
R+1:PRINT D$
1070 RETURN
```

Nu moet u vooral niet menen, dat het allemaal in één keer op deze manier werd ondergebracht in het programma. Welnee, Zo werkt dat niet in de praktijk. Er moeten in een programmerend mens altijd eerst diverse lichtjes opgaan, alvorens hij zich bondig weet uit te drukken. Je kunt een hele tijd over de meest simpele oplossing heen kijken. Het blijft een hobby. Niet al te veel lieden hameren beroepshalve rond op hun MSX. En al kun je zo'n oplossing wellicht nooit nog eens gebruiken, handigheidjes zullen desondanks genoteerd worden in het dagboek van de MSXer. Men kan nooit weten.

Tussendoor bent u weer eens aan het typen gezet. De print-routine, opgeroepen in regel 1040. Het is natuurlijk ook anders te formuleren dan hier is gedaan. Vooral niet nalaten als het korter kan zijn. De enige C waarmee we werken in het programma is de eerste kode van een schaakstuk. Dat hangt samen met de lokatiewaarden van een sprite. Daarvoor geldt altijd het punt linksboven. En daar zit aldus onze eerste kode vlak in de buurt. Vandaar. Alleen in de printroutine moeten ook de overige 3 kodes boven water komen. In dezelfde volgorde als de 8x8 matrixes van een 16x16 sprite in de patroonadressen van BASE (9) worden geladen. Die K en R hadden we ons al verschaft in regel 440 van de schermleesroutine. Daarmee is het allemaal wel gezegd aangaande deze printroutine. GOSUB 1430 in regel 800 kennen we ook al zodra het stuk is gePRINT. Zo komen we aan de laatste GOSUB. De promotiekontrole.

```
1450 REM ** promotie **
1460 IF R=2 OR R=16 THEN 1470 ELSE RETUR
N
```

```
1470 K=INT(X/8):R=INT(Y/8)+1
1480 O=BASE(5):C=VPEEK(O+(R*32+K))
1490 IF C=128 THEN 1530
1500 IF C=152 THEN 1540
1510 IF C=176 THEN 1560
1520 IF C=200 THEN 1570 ELSE RETURN
1530 C=144:GOTO 1550
1540 C=168
1550 GOSUB 1040:GOSUB 1360:GOSUB 1240:GO
SUB 1180:Z=4:GOTO 1990
1560 C=192:GOTO 1580
1570 C=216
1580 GOSUB 1040:GOSUB 1360:GOSUB 1230:GO
SUB 1180:Z=4:GOTO 1990
```

Andermaal getypt? Het mag u een troost zijn, dat dit nog slechts éénmaal hoeft te gebeuren. Maar dan wel wat meer regels. De sprong naar deze promotieroutine vergt normaal amper tijd. Zo vaak gebeurt het niet, dat een pion op de koningslijn van de vijand weet door te dringen. Weliswaar mag de gelukkige dan zichzelf inruilen voor elk ander stuk behalve de koning, maar vrijwel altijd zal voor de dame worden gekozen. Schakers zijn heel geëmancipeerd.
Schaaksters in dit geval wat éénkennig. Daarom betekent promotie hier die in een dame of koningin. In regel 1460 wordt onnodige lezing van de routine voorkomen. R=2 en R=16 betreffende koningslijnen. Als R een andere waarde heeft, dan kan meteen rechtsomkeerd worden gemaakt. En al kent R de waarde 2 of 16, dan nog zal eerst moeten worden vastgesteld in hoeverre het gaat om een pion met promotie-ambities. Dan doen we dus weer via VPEEK. Betreft het inderdaad een pion, dan begint de werkelijke aktie. De kode van de pion verandert in die van de dame in dezelfde kleurenkombinatie en Z ontvangt de waarde 4 zoals verbonden met de dame. Dat is nu niet nodig voor de sprite. GOSUB 1040 in regel 1550 maakt duidelijk, dat die dame meteen in de plaats van de pion wordt gePRINT. Als de bijbehorende teksten op het scherm zijn geciteerd, dan volgt er in plaats van RETURN een GOTO. Daarvan is regelnummer 1990 er eentje van de schaakroutine. Die hele routine hoeft niet te worden gelezen. Er kan vanuit de promotieroutine meteen naar die voor de dame worden gewipt. Zou tijdens de promotieplechtigheid de vijandige koning op die lijn aanwezig zijn zonder bescherming van een gelijkgezind stuk, dan komt zijne majesteit grandioos schaak te staan. En zo behoort het

ook te worden aangegeven in het tekstkader onder het schaakbord. Maar die regel 1990 komen we vanzelf weer tegen, als we ons met het allerlaatste onderdeel van het programma gaan bezighouden. Daarvoor keren we terug naar het hoofdprogramma.

Wanneer zwart of wit beide akties ten uitvoer hebben gebracht, dan komen we in regel 200 terecht. Opgemerkt werd reeds, dat elke plaatsverandering van de koning terwille van de schaakroutine bijgehouden moet worden. Daarvoor hebben we XP en PX verzonnen. Die staan steeds voor de aktuele positie van de koningen. Maar ook als de koning niet wordt verplaatst, dan wordt evengoed met GOTO 1850 naar de schaakroutine gegaan.
In het begin is dat meestal overbodig. Het is evenwel niet vast te stellen na hoeveel zetten iemand de eerste schaaksituatie veroorzaakt. Het lukt sommigen al na enkele zetten om de ander schaakmat te zetten. Mijn zoon kende die opening en ik ben nog steeds niet over de frustratie heen, al is het intussen een honderd jaar geleden traumatisch geworden. Daarom moeten we vanaf de eerste zet de schaakroutine laten lezen. U verwacht nu het laatste typwerk. Het is echter wijzer om in dit stadium van hoofdstuk 10 mede ter voorbereiding daarop aandacht te schenken aan nog eens de schermlezing. Nu echter niet om bepaalde kodes te verkrijgen. Hoe dat te realiseren is, dat weten we nu langzamerhand wel. U hebt u al voorbereid om het in uw eigen programma toe te passen. De vakjes van het schaakbord, althans het blokje linksboven daarvan, vertegenwoordigen niet alleen de beginkodes van schaakstukken en kleuren. Ze hebben óók een eigen lokatienummer in de adressen van BASE (5).

Schermposities

	a	b	c	d	KONING	f	g	h
8	66	68	70	72	74	76	78	80
7	130	132	134	136	138	140	142	144
6	194	196	198	200	202	204	206	208
5	258	260	262	264	266	268	270	272
4	322	324	326	328	330	332	334	336
3	386	388	390	392	394	396	398	400
2	450	452	454	456	458	460	462	464
1	514	516	518	520	522	524	526	528
					KONING			

Zo kunnen we ons deze lokaties voorstellen in het schaakbord. Hoe komen we aan die lokatienummers?

K=INT (X/8):R=INT (Y/8)+ 1
S=BASE (5) + (R*32+K)-6144

De S staat voor een lokatienummer. Omdat we niet met grote getallen willen werken, trekken we het startadres van BASE (5), 6144, er meteen weer vanaf. Zo houden we er waarden aan over, waarmee te werken is. Bijvoorbeeld:

A=(S-28) /2

De hoogste waarde 528 komt daarmee op 250 en de laagste, 66, op 38. Nou en? Nu, daarmee blijven we binnen de kodenummers van de karaktersets. Dat maakt het mogelijk, om deze lokatienummers om te vormen tot kodenummers, waarvan de bijbehorende karakters in een speelstring kunnen worden gezet. En daarmee zou weer een bepaalde zet kunnen worden uitgevoerd óf gekontroleerd. Immers, een string kunnen we binnen een lus laten lezen.

```
FOR I=1 TO 4
B=ASC (MID$ (S$, I,1))
NEXT I
```

Die S$ speelstring zou er zo uit kunnen zien:

```
S$="⟨D é"
```

Deze simpele string is in de schaakroutine gebruikt. Een speelstring kan echter even goed uit 64 karakters bestaan. Zoveel als er aan lokatienummers binnen het schaakbord aanwezig zijn. Daarbij is uiteraard ook een lus van 1 tot 64 nodig. Krijgt u het door? Kombineren we zo'n komplete speelstring namelijk met een tweede string, waaruit de kodewaarden voor een te printen schaakstuk kunnen worden gehaald, dan resten ons in zo'n lus alleen nog de nodige voorwaarderegels om de computer zelf te laten reageren op onze zet en die met een tegenzet te beantwoorden. Omdat er een verscheidenheid aan openingszetten bestaat, zal de mogelijkheid moeten worden ingebouwd om uit diverse speelstrings te laten kiezen. Het ligt er dan maar aan in welke volgorde wij de karakters in zo'n string onderbrengen. De meest essentiële zetten zullen daarbij het eerst gelezen moeten woren. Zijn ze door de spelsituatie of de voorwaarderegels niet uitvoerbaar, dan wordt de volgende zetmogelijkheid uit de string gelezen. Zolang totdat een zet kan volgen. Eenvoudiger is op deze wijze een komplete zetkontrole door te voeren. Het stuk dat slechts op één manier te verplaatsen is, heeft genoeg aan één string en een ander stuk, de dame bijvoorbeeld, zal er twee moeten krijgen.

Het gaat ook hierbij niet uitsluitend om het schaakbord. Middels speelstrings kunnen andere bordspellen evenzeer gespeeld worden. Het gaat om de methode. U moet die alleen zelf uitwerken. Dit programma leent zich er goed voor, om een en ander alvast in toe te passen. Op heel bescheiden schaal is zo'n string aldus ook in de schaakroutine toegepast. En dat betekent het laatste typwerk. Althans wat ons programma betreft.

```
1810 REM ** positie koning **
1820 XP=BASE(5)+(R*32+K)-6144:RETURN
1830 PX=BASE(5)+(R*32+K)-6144:RETURN
1840 REM ** schaak situaties **
1850 S=BASE(5)+(R*32+K)-6144
1860 IF Z=0 OR Z=2 THEN 1900
1870 IF Z=3 OR Z=4 THEN 1990
1880 IF Z=1 OR Z=4 THEN 2070
1890 GOTO 250
1900 PD$="<D~é":PN$="B>"
1910 FOR I=1 TO 4
1920 PRD=ASC(MID$(PD$,I,1))
1930 IF I<3 THEN 1940 ELSE 1950
1940 NP=ASC(MID$(PN$,I,1))
1950 IF SP=1 THEN 1960 ELSE 1970
1960 IF S=PX+NP OR S=PX+PRD OR S=PX-PRD
THEN 2160 ELSE 1980
1970 IF S=XP-NP OR S=XP+PRD OR S=XP-PRD
THEN 2160
1980 NEXT I:GOTO 250
1990 REM ** loper/dame/toren **
2000 IF SP=1 THEN YY=PX ELSE IF SP=0 THE
N YY=XP
2010 IF S>YY THEN 2040 ELSE 2020
2020 FOR I=S TO YY STEP 62:GOSUB 2100:NE
XT I
2030 FOR I=S TO YY STEP 66:GOSUB 2100:NE
XT I:GOTO 2060
2040 FOR I=S TO YY STEP -62:GOSUB 2100:N
EXT I
2050 FOR I=S TO YY STEP -66:GOSUB 2100:N
EXT I
2060 IF Z=4 THEN 2070 ELSE 250
2070 FOR I=S TO YY STEP 2:GOSUB 2100:NEX
T I
2080 FOR I=S TO YY STEP -2:GOSUB 2100:NE
XT I
```

```
2090 GOTO 250
2100 ZK=VPEEK(I+6144)
2110 IF ZK=224 OR ZK=232 THEN 2150 ELSE
2120
2120 IF SP=1 THEN 2130 ELSE 2140
2130 IF ZK=148 OR ZK=172 THEN 2160 ELSE
2150
2140 IF ZK=196 OR ZK=220 THEN 2160 ELSE
2150
2150 RETURN
2160 REM ** schaak **
2170 GOSUB 1360
2180 G=G+1:BEEP:FOR I=1 TO 50:NEXT I:LOC
ATE 2,20:PRINT Y$:FOR I=1 TO 50:NEXT I:L
OCATE 7,20:PRINT"SCHAAK!"
2190 IF G<10 THEN 2180
2200 G=0:GOTO 250
```

In de regels 1820 en 1830 komt u die bij te houden XP en YP waarden tegen voor de positie van de koningen. En dat is het nummer van de schermlokatie. Eenzelfde nummer wordt in S gegeven aan het schaakstuk, dat mogelijkerwijs een koning schaak kan geven. Door de regels 1860 tot en met 1890 wordt gericht gezocht naar zo'n situatie. De schaakstukken hebben daarmee hun eigen routines gekregen. Ofschoon dit snel verwerkbaar is, kent het alsnadeel, dat alleen direkte schaaksituaties worden gemeld. Wanneer door het verplaatsen van een stuk daardoor een ander stuk de koning schaak heeft, dan zal er meestal geen schaakmelding komen. Evenmin is er een herhaling mogelijk van een schaakmelding in het geval de eerste door biede spelers zou worden genegeerd.

De beide strings staan in regel 1900. Het gaat hier om paard en pion. Het is in dezelfde lus ondergebracht. De waarde in PRD en NP worden bij die van de koninglokatie geteld. Is de som gelijk aan de S-waarde van het betreffende schaakstuk, dan is er sprake van een schaaksituati. U moet die vooral zelf eens gaan uitspitten aan de hand van de weergegeven schermlokaties. Des te duidelijker zal het worden, welke mogelijkheden de speelstring-methode kan bieden. Vanuit de promotieroutine werd verwezen naar regel 1990 en die hebben we hier dan. Deze routine is anders gekonstrueerd. Eerst wordt één variabele, YY, genomen voor XP of PX. Dat bespaart een extra routine.

Voor de diagonale zetten van dame en loper is per lokatie een verschil van 62 of 64 te konstateren in de schermlokatie-grafiek. Gedacht vanuit de koning naar boven of naar beneden. Voor dame en toren bedraagt dit verschil per lokatie 2. Eén en ander is ondergebracht in de lussen. Voordien vindt er een selektie plaats in regel 2010. In de lussen staat alleen de GOSUB, die naar de eigenlijke routine verwijst. Hier moet opnieuw de kode van een lokatie worden gezocht, zoals deze in opeenvolging door de I-lussen wordt bepaald. Daarvoor dient uiteraard de eerst afgetrokken waarde van het startadres van BASE (5) weer bij die I te worden geteld. Anders zou er geVPEEKed worden in de laagste adressen van VRAM en daar zijn de karakterpatronen ondergebracht. Wordt zo de kode voor een leeg vakje tegengekomen, dan kan meteen via RETURN terug gegaan worden naar de lus. Is het evenwel de kode van een schaakstuk, dan moet bepaald worden of het daarbij om de koning gaat. Als dit juist is, dan is aldus een schaaksituatie ontstaan en volgt daarvan de melding op het scherm. Betreft het een ander stuk, dan wordt de lus opnieuw doorlopen.

Voor zo'n schaakkontrole zijn meerdere formuleringen te bedenken. Al werkt deze goed, de meest ideale is het zeker niet. Daarom valt er ook hier zo het één en ander te knutselen aan het programma. Het allerlaatste is hier de routine, waarin van een schaaksituatie melding wordt gemaakt. Het is een manier om een tekst enige malen afwisselend te PRINTen en te wissen met een begeleidend BEEP-toontje. Zo'n knippertekst zal speciaal gebruikt worden, als er extra aandacht voor iets wordt gevraagd. En dat mag bij een schaakmelding het geval worden genoemd.

Het programma dat u zeer binnekort zelf gaat schrijven, zal er heel anders uitzien, dan dit schaakbordprogramma. Maar u komt er wél vergelijkbare zaken in tegen. En er zal eveneens bij moeten worden nagedacht hoe iets tot stand kan worden gebracht met ook alle zorgvuldigheid vandien bij het gebruik van variabelen en waarden. Niet te vergeten ook de logische konstruktie van het programma. De uitvoerigheid waarmee aandacht gegeven is aan het schaakbordprogramma zal er aan kunnen bijdragen, dat u voortaan gerichter of wellicht zelfs voor de eerste maal echt serieus aan de slag kunt gaan. En nu u weet, hoe VPOKE en VPEEK gebruikt kunnen worden, kunt u de aandacht eisende zaken aan uw programma toevoegen. Anderen zullen zich wellicht het hoofd breken over het hoe daarvan. U niet meer.

Appendix 1
MSX karaktersets

MSX-karaktersets

De verdeling in 32 sets met elk 8 karakters komt niet terug in Basic-instrukties en blijft doorgaans ongenoemd. Kleurwijziging van karakters is echter alleen per set mogelijk. Het is weinig zinvol om daarvoor vanaf een willekeurig kodenummer te beginnen. Ook bij het wijzigen van een karakter in een zelf ontworpen figuurtje is het logischer om met het eerste kodenummer van een set te beginnen.

Volgens de standaard ASCII-sets (de nummers 5 t/m 16) behoort kode 127 leeg te zijn; aangeduid met DELete. Bij sommige MSX-computers is aan kode 127 wél een karakter toegekend en is kode 201 leeg gebleven. Om die reden zijn bij deze kodes geen toetskombinaties vermeld. En uiteraard evenmin bij de ASCII-sets. Voor MSX is het belangrijkste, dat alle karakters aanwezig zijn. Zo is het mogelijk, dat die karakters niet op alle computers volgens de aangegeven toetskombinaties op het scherm te zetten zijn. Het is verstandig, om dit voor uw computer te kontroleren. Het beste kan dit door zonder regelnummer in te typen:

PRINT CHR$ (kodenummer)

Na RETURN wordt dan het bij het vermelde kodenummer behorende karakter weergegeven. Door de aangegeven toetskombinatie in te drukken, is dan vast te stellen of deze ook de juiste is voor uw MSX. Algemeen zal dit korrekt zijn. Is dit niet het geval, dan kunt u het vrije deel van de toetskadertjes benutten door daarin aan te geven, welke toetsen wél in te drukken zijn. Wilt u het helemaal goed doen, dan kunt u met wrijfletters in het betreffende kadertje een pijltje zetten en het bestaande pijltje afdekken met bijvoorbeeld korrektielak. Een velletje pijltjes kost een paar gulden en is bij de gespecialiseerde kantoorboekhandel verkrijgbaar. Voor de kodes lager dan 32 geldt een andere formule:

PRINT CHR$(1) ; CHR$(65)

Voor kodenummer 2 wordt (65) met 1 verhoogd tot en met 95 voor kode 31; (1) blijft steeds gelijk.

De MSX-karaktersets vormen een belangrijk bestanddeel van de basisinformatie, waarover u dient te beschikken. Voor zowel tekstverwerking als bij wijziging van kleur en patroon zult u deze sets regelmatig raadplegen. Zo is er wat voor te zeggen, om deze lijsten te kopiëren op stevig papier en bijvoorbeeld in een hoesje bij de hand te houden. Tegen het herhaalde raadplegen van de lijsten is op de duur ook het best ingebonden boek niet bestand.

MSX - karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
1	0						
	1		↓			↓	[
	2		↓	↓		↓	[
	3		↓	↓		↓	"
	4		↓	↓		↓	;
	5		↓			↓	"
	6		↓			↓	;
	7		↓			↓	9
2	8		↓	↓		↓	9
	9		↓			↓	0
	10		↓	↓		↓	0
	11		↓			↓	M
	12		↓	↓		↓	M
	13		↓			↓	]
	14		↓	↓		↓	]
	15		↓			↓	Z

MSX – karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
3	16		↓			↓	G
	17		↓			↓	B
	18		↓			↓	T
	19		↓			↓	H
	20		↓			↓	F
	21		↓	↓		↓	G
	22		↓	↓		↓	\
	23		↓			↓	–
4	24		↓			↓	R
	25		↓			↓	Y
	26		↓			↓	V
	27		↓			↓	N
	28		↓			↓	X
	29		↓			↓	/
	30		↓			↓	\
	31		↓	↓		↓	–

MSX - karakter sets

SET	CODE	KARAKTER
5	32	spatie
	33	
	34	
	35	
	36	
	37	
	38	
	39	

SET	CODE	KARAKTER
6	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	

SET	CODE	KARAKTER
7	48	
	49	
	50	
	51	
	52	
	53	
	54	
	55	

SET	CODE	KARAKTER
8	56	
	57	
	58	
	59	
	60	
	61	
	62	
	63	

SET	CODE	KARAKTER
9	64	
	65	
	66	
	67	
	68	
	69	
	70	
	71	

SET	CODE	KARAKTER
10	72	
	73	
	74	
	75	
	76	
	77	
	78	
	79	

MSX - karakter sets

SET	CODE	KARAKTER
11	80	P
	81	Q
	82	R
	83	S
	84	T
	85	U
	86	V
	87	W
12	88	X
	89	Y
	90	Z
	91	[
	92	\
	93	]
	94	^
	95	_
13	96	`
	97	a
	98	b
	99	c
	100	d
	101	e
	102	f
	103	g
14	104	h
	105	i
	106	j
	107	k
	108	l
	109	m
	110	n
	111	o
15	112	p
	113	q
	114	r
	115	s
	116	t
	117	u
	118	v
	119	w
16	120	x
	121	y
	122	z
	123	{
	124	|
	125	}
	126	~
	127	delete

MSX - karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
17	128		↓		↓		9
	129				↓		g
	130				↓		u
	131				↓		q
	132				↓		a
	133				↓		z
	134				↓		‹
	135				↓		9
18	136				↓		w
	137				↓		s
	138				↓		x
	139				↓		d
	140				↓		e
	141				↓		c
	142		↓		↓		A
	143		↓		↓		‹

MSX - karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
19	144		↓		↓		U
	145				↓		J
	146		↓		↓		J
	147				↓		R
	148				↓		F
	149				↓		V
	150				↓		T
	151				↓		B
20	152				↓		5
	153		↓		↓		F
	154		↓		↓		G
	155		↓		↓		4
	156		↓	↓	↓		4
	157		↓	↓	↓		5
	158		↓	↓	↓		2
	159		↓		↓		1

MSX – karakter sets

SET	CODE	KARAKTER	TOETSEN (↓ is samen indrukken) CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
21	160				↓		y
	161				↓		i
	162				↓		o
	163				↓		p
	164				↓		n
	165		↓		↓		N
	166		↓		↓		>
	167		↓		↓		/
22	168		↓	↓	↓		/
	169		↓	↓		↓	R
	170		↓	↓		↓	Y
	171		↓			↓	2
	172		↓			↓	1
	173		↓	↓	↓		1
	174		↓	↓		↓	<
	175		↓	↓		↓	>

MSX - karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
23	176		↓		↓		H
	177				↓		h
	178		↓		↓		K
	179				↓		k
	180		↓		↓		L
	181				↓		L
	182		↓		↓		;
	183				↓		;
24	184		↓		↓		"
	185				↓		"
	186		↓			↓	3
	187		↓			↓	~
	188		↓			↓	C
	189		↓			↓	5
	190		↓	↓	↓		3
	191		↓		↓		3

MSX - karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
25	192		↓			↓	U
	193		↓	↓		↓	D
	194		↓			↓	O
	195		↓	↓		↓	O
	196		↓			↓	A
	197		↓	↓		↓	U
	198		↓			↓	J
	199		↓			↓	D
26	200		↓			↓	L
	201						
	202		↓	↓		↓	J
	203		↓	↓		↓	Q
	204		↓			↓	Q
	205		↓			↓	E
	206		↓	↓		↓	E
	207		↓			↓	W

MSX - karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
27	208		↓	↓		↓	W
	209		↓	↓		↓	S
	210		↓			↓	S
	211		↓	↓		↓	N
	212		↓	↓		↓	F
	213		↓	↓		↓	V
	214		↓	↓		↓	H
	215		↓	↓		↓	P
28	216		↓	↓	↓		0
	217		↓		↓		2
	218		↓		↓		]
	219		↓			↓	P
	220		↓			↓	I
	221					↓	k
	222		↓	↓		↓	K
	223		↓	↓		↓	I

MSX - karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
29	224		↓		↓		6
	225		↓		↓		7
	226		↓	↓	↓		8
	227		↓	↓	↓		P
	228		↓	↓	↓		~
	229		↓		↓		~
	230		↓		↓		M
	231		↓		↓		8
30	232		↓	↓	↓		[
	233		↓		↓		=
	234		↓	↓	↓		]
	235		↓		↓		0
	236		↓			↓	8
	237		↓		↓		[
	238		↓		↓		_
	239		↓			↓	4

MSX - karakter sets

TOETSEN (↓ is samen indrukken)

SET	CODE	KARAKTER	CAPS LOCK	SHIFT	CODE	GRAPH	TOETS
31	240		↓	↓		↓	=
	241		↓			↓	=
	242		↓			↓	>
	243		↓			↓	<
	244		↓			↓	6
	245		↓	↓		↓	6
	246		↓	↓		↓	/
	247		↓	↓		↓	~
32	248		↓	↓		↓	Z
	249		↓	↓		↓	X
	250		↓	↓		↓	C
	251		↓			↓	7
	252		↓	↓		↓	3
	253		↓	↓		↓	2
	254		↓	↓		↓	A
	255		cursor				

appendix 2
Patroontabellen

De 256 manieren om een byte nader te definiëren in vooral VRAM zijn weergegeven met de meest gebruikte kodering in decimaal en hexadecimaal. De hex-kode zal algemeen de voorkeur kunnen hebben. Het bestaat uit slechts 2 getallen; A-B-C-D-E- en F vervangen hierin de getallen 10 t/m 16. Omdat hiervoor de 8 bytes gesplitst worden in 2 maal 4 bits en zo'n helft slechts op 16 manieren van een patroon is te voorzien, is de hex-kode tegelijk het gemakkelijkste te onthouden. Voor de decimale waarden moeten de lijsten telkens weer geraadpleegd worden. Vooral in lussen van bijvoorbeeld adressen en aan variabelen zal eerder de decimale waarde worden toegekend. Algemeen is het geen bezwaar, wanneer beide kodes door elkaar worden gebruikt.

Dec.	Hex.
0	&H0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	&H10
17	11
18	12
19	13
20	14
21	15
22	16
23	17

Dec.	Hex.
24	&H18
25	19
26	1A
27	1B
28	1C
29	1D
30	1E
31	1F
32	&H20
33	21
34	22
35	23
36	24
37	25
38	26
39	27
40	28
41	29
42	2A
43	2B
44	2C
45	2D
46	2E
47	2F

Dec.	Hex.
48	&H30
49	31
50	32
51	33
52	34
53	35
54	36
55	37
56	38
57	39
58	3A
59	3B
60	3C
61	3D
62	3E
63	3F
64	&H40
65	41
66	42
67	43
68	44
69	45
70	46
71	47

Dec.	Hex.
72	48
73	49
74	4A
75	4B
76	4C
77	4D
78	4E
79	4F
80	&H50
81	51
82	52
83	53
84	54
85	55
86	56
87	57
88	58
89	59
90	5A
91	5B
92	5C
93	5D
94	5E
95	5F

Dec.	Hex.
96	60
97	61
98	62
99	63
100	64
101	65
102	66
103	67
104	68
105	69
106	6A
107	6B
108	6C
109	6D
110	6E
111	6F
112	&H70
113	71
114	72
115	73
116	74
117	75
118	76
119	77

Dec.	Hex.
120	78
121	79
122	7A
123	7B
124	7C
125	7D
126	7E
127	7F
128	&H80
129	81
130	82
131	83
132	84
133	85
134	86
135	87
136	88
137	89
138	8A
139	8B
140	8C
141	8D
142	8E
143	8F

Dec.	Hex.
144	&H90
145	91
146	92
147	93
148	94
149	95
150	96
151	97
152	98
153	99
154	9A
155	9B
156	9C
157	9D
158	9E
159	9F
160	&HA0
161	A1
162	A2
163	A3
164	A4
165	A5
166	A6
167	A7

Dec.	Hex.
168	A8
169	A9
170	AA
171	AB
172	AC
173	AD
174	AE
175	AF
176	&HB0
177	B1
178	B2
179	B3
180	B4
181	B5
182	B6
183	B7
184	B8
185	B9
186	BA
187	BB
188	BC
189	BD
190	BE
191	BF

Dec.	Hex.
192	&HC0
193	C1
194	C2
195	C3
196	C4
197	C5
198	C6
199	C7
200	C8
201	C9
202	CA
203	CB
204	CC
205	CD
206	CE
207	CF
208	&HD0
209	D1
210	D2
211	D3
212	D4
213	D5
214	D6
215	D7

Dec.	Hex.
216	D8
217	D9
218	DA
219	DB
220	DC
221	DD
222	DE
223	DF
224	&HE0
225	E1
226	E2
227	E3
228	E4
229	E5
230	E6
231	E7
232	E8
233	E9
234	EA
235	EB
236	EC
237	ED
238	EE
239	EF

Dec.	Hex.
240	&HF0
241	F1
242	F2
243	F3
244	F4
245	F5
246	F6
247	F7
248	F8
249	F9
250	FA
251	FB
252	FC
253	FD
254	FE
255	FF

appendix 3
Kleurtabellen

Kleurwijziging van karakters in de MSX-sets is alleen mogelijk per set van 8 karakters. Zie APPENDIX-1. Begonnen wordt met de 1e karakterkode van een set. Deze kleurwijziging middels VPOKE(BASE(6) heeft prioriteit boven COLOR. De kleurwaarden zijn zowel in decimaal als hex. gegeven.

kleur	kode	&H	kleur	kode	&H
transparant	0	0	middelrood	8	8
zwart	1	1	lichtrood	9	9
middelgroen	2	2	donkergeel	10	A
lichtgroen	3	3	lichtgeel	11	B
donkerblauw	4	4	donkergroen	12	C
lichtblauw	5	5	magenta	13	D
donkerrood	6	6	grijs	14	E
cyaan	7	7	wit	15	F

In de tabellen is het kleurkontrast aangegeven.
Gebruikte afkortingen: A - voorgrondkleur
B - achtergrondkleur
1 - goed kontrast; bij voorkeur gebruiken
2 - matig kontrast
3 - slecht kontrast; niet gebruiken

MSX-kleurtabellen

kleur A - B	kode &H A - B	decim.	kontrast	kleur A - B	kode &H A - B	decim.	kontrast
0 0	00	0		1 0	10	16	1
0 1	01	1	1	1 1	11	17	1
0 2	02	2	1	1 2	12	18	1
0 3	03	3	1	1 3	13	19	1
0 4	04	4	1	1 4	14	20	1
0 5	05	5	1	1 5	15	21	1
0 6	06	6	1	1 6	16	22	1
0 7	07	7	1	1 7	17	23	1
0 8	08	8	1	1 8	18	24	1
0 9	09	9	1	1 9	19	25	1
0 10	0A	10	1	1 10	1A	26	1
0 11	0B	11	1	1 11	1B	27	1
0 12	0C	12	1	1 12	1C	28	1
0 13	0D	13	1	1 13	1D	29	1
0 14	0E	14	1	1 14	1E	30	1
0 15	0F	15	1	1 15	1F	31	1
2 0	20	32	2	3 0	30	48	2
2 1	21	33	1	3 1	31	49	1
2 2	22	34	1	3 2	32	50	2
2 3	23	35	2	3 3	33	51	1
2 4	24	36	2	3 4	34	52	1
2 5	25	37	3	3 5	35	53	2
2 6	26	38	3	3 6	36	54	1
2 7	27	39	2	3 7	37	55	3
2 8	28	40	3	3 8	38	56	2
2 9	29	41	3	3 9	39	57	3
2 10	2A	42	3	3 10	3A	58	3
2 11	2B	43	3	3 11	3B	59	3
2 12	2C	44	2	3 12	3C	60	1
2 13	2D	45	3	3 13	3D	61	2
2 14	2E	46	3	3 14	3E	62	3
2 15	2F	47	1	3 15	3F	63	2
4 0	40	64	1	5 0	50	80	2
4 1	41	65	1	5 1	51	81	1
4 2	42	66	3	5 2	52	82	3
4 3	43	67	2	5 3	53	83	2
4 4	44	68	1	5 4	54	84	1
4 5	45	69	1	5 5	55	85	1
4 6	46	70	3	5 6	56	86	2
4 7	47	71	1	5 7	57	87	2
4 8	48	72	1	5 8	58	88	3
4 9	49	73	1	5 9	59	89	1
4 10	4A	74	1	5 10	5A	90	2
4 11	4B	75	1	5 11	5B	91	3
4 12	4C	76	3	5 12	5C	92	3
4 13	4D	77	1	5 13	5D	93	3
4 14	4E	78	1	5 14	5E	94	1
4 15	4F	79	1	5 15	5F	95	1

kleur A - B	kode &H A - B	decim.	kontrast	kleur A - B	kode &H A - B	decim.	kontrast
6 0	60	96	3	7 0	70	112	1
6 1	61	97	1	7 1	71	113	2
6 2	62	98	3	7 2	72	114	3
6 3	63	99	2	7 3	73	115	1
6 4	64	100	2	7 4	74	116	3
6 5	65	101	2	7 5	75	117	1
6 6	66	102	1	7 6	76	118	2
6 7	67	103	2	7 7	77	119	1
6 8	68	104	1	7 8	78	120	2
6 9	69	105	1	7 9	79	121	3
6 10	6A	106	1	7 10	7A	122	3
6 11	6B	107	1	7 11	7B	123	2
6 12	6C	108	3	7 12	7C	124	1
6 13	6D	109	1	7 13	7D	125	2
6 14	6E	110	1	7 14	7E	126	3
6 15	6F	111	1	7 15	7F	127	1
8 0	80	128	1	9 0	90	144	1
8 1	81	129	1	9 1	91	145	1
8 2	82	130	3	9 2	92	146	3
8 3	83	131	2	9 3	93	147	3
8 4	84	132	1	9 4	94	148	1
8 5	85	133	3	9 5	95	149	1
8 6	86	134	1	9 6	96	150	1
8 7	87	135	2	9 7	97	151	3
8 8	88	136	1	9 8	98	152	1
8 9	89	137	1	9 9	99	153	1
8 10	8A	138	1	9 10	9A	154	3
8 11	8B	139	1	9 11	9B	155	2
8 12	8C	140	3	9 12	9C	156	2
8 13	8D	141	3	9 13	9D	157	1
8 14	8E	142	1	9 14	9E	158	3
8 15	8F	143	1	9 15	9F	159	1
10 0	A0	160	1	11 0	B0	176	1
10 1	A1	161	1	11 1	B1	177	1
10 2	A2	162	3	11 2	B2	178	1
10 3	A3	163	3	11 3	B3	179	2
10 4	A4	164	1	11 4	B4	180	1
10 5	A5	165	2	11 5	B5	181	1
10 6	A6	166	1	11 6	B6	182	1
10 7	A7	167	3	11 7	B7	183	2
10 8	A8	168	1	11 8	B8	184	1
10 9	A9	169	2	11 9	B9	185	2
10 10	AA	170	1	11 10	BA	186	3
10 11	AB	171	2	11 11	BB	187	1
10 12	AC	172	1	11 12	BC	188	1
10 13	AD	173	2	11 13	BD	189	2
10 14	AE	174	3	11 14	BE	190	3
10 15	AF	175	1	11 15	BF	191	1

kleur A - B	kode &H A - B	decim.	kontrast	kleur A - B	kode &H A - B	decim.	kontrast
12 0	C0	192	3	13 0	D0	208	1
12 1	C1	193	1	13 1	D1	209	1
12 2	C2	194	3	13 2	D2	210	3
12 3	C3	195	1	13 3	D3	211	3
12 4	C4	196	3	13 4	D4	212	1
12 5	C5	197	3	13 5	D5	213	3
12 6	C6	198	3	13 6	D6	214	2
12 7	C7	199	1	13 7	D7	215	2
12 8	C8	200	3	13 8	D8	216	3
12 9	C9	201	3	13 9	D9	217	1
12 10	CA	202	1	13 10	DA	218	2
12 11	CB	203	1	13 11	DB	219	1
12 12	CC	204	1	13 12	DC	220	3
12 13	CD	205	3	13 13	DD	221	1
12 14	CE	206	1	13 14	DE	222	1
12 15	CF	207	1	13 15	DF	223	1
14 0	E0	224	1	15 0	F0	240	1
14 1	E1	225	1	15 1	F1	241	1
14 2	E2	226	1	15 2	F2	242	1
14 3	E3	227	3	15 3	F3	243	1
14 4	E4	228	1	15 4	F4	244	1
14 5	E5	229	1	15 5	F5	245	1
14 6	E6	230	1	15 6	F6	246	1
14 7	E7	231	3	15 7	F7	247	1
14 8	E8	232	1	15 8	F8	248	1
14 9	E9	233	2	15 9	F9	249	1
14 10	EA	234	3	15 10	FA	250	1
14 11	EB	235	3	15 11	FB	251	1
14 12	EC	236	1	15 12	FC	252	1
14 13	ED	237	1	15 13	FD	253	1
14 14	EE	238	1	15 14	FE	254	1
14 15	EF	239	1	15 15	FF	255	1

appendix 4
Tokenlijst

De computer zet de Basic-instrukties zoals PRINT om in kodes, aangeduid als token. De Basic-instrukties met hun tokennummer zijn ondergebracht in de sleutelwoordtabel in de adressen 14960 t/m 15673 in alfabetische volgorde. In de weergegeven tabel is de numerieke volgorde gebruikt in decimalen. Tokens zijn niet van belang bij het programmeren in Basic. Wél bij gebruik van machinekode. Voor het lezen van RAM-adressen waarin Basic regels zijn opgeslagen kunnen de nummers geraadpleegd worden.

MSX-tokenlijst

nr.	instruktie	nr.	instruktie	nr.	instruktie	nr.	instruktie
1	LEFT$	50		99		147	LIST
2	RIGHT$	51		100		148	NEW
3	MID$	52		101		149	ON
4	SGN	53		102		150	WAIT
5	INT(erval)	54		103		151	DEF
6	ABS	55		104		152	POKE
7	SQR	56		105		153	CONT
8	RND	57		106		154	CSAVE
9	SIN	58		107		155	CLOAD
10	LOG	59		108		156	OUT
11	EXP	60		109		157	LPRINT
12	COS	61		110		158	LLIST
13	TAN	62		111		159	CLS
14	ATN	63		112		160	WIDTH
15	FRE	64		113		161	ELSE
16	INP	65		114		162	TRON
17	POS	66		115		163	TROFF
18	LEN	67		116		164	SWAP
19	STR$	68		117		165	ERASE
20	VAL	69		118		166	ERROR
21	ASC	70		119		167	RESUME
22	CHR$	71		120		168	DELETE
23	PEEK	72		121		169	AUTO
24	VPEEK	73		122		170	RENUM
25	SPACE$	74		123		171	DEFSTR
26	OCT$	75		124		172	DEFINT
27	HEX$	76		125		173	DEFSNG
28	LPOS	77		126		174	DEFDBL
29	BIN$	78		127		175	LINE
30	CINT	79		128		176	OPEN
31	CSNG	80		129	END	177	FIELD
32	CDBL	81		130	FOR	178	GET
33	FIX	82		131	NEXT	179	PUT
34	STICK	83		132	DATA	180	CLOSE
35	STRIG	84		133	INPUT	181	LOAD
36	PDL	85		134	DIM	182	MERGE
37	PAD	86		135	READ	183	FILES
38	DSKF	87		136	LET	184	LSET
39	FPOS	88		137	GOTO/ GO TO	185	RSET
40	CVI	89				186	SAVE
41	CVS	90		138	RUN	187	LFILES
42	CVD	91		139	IF	188	CIRCLE
43	EOF	92		140	RESTORE	189	COLOR
44	LOC	93		141	GOSUB	190	DRAW
45	LOF	94		142	RETURN	191	PAINT
46	MKI$	95		143	REM	192	BEEP
47	MKS$	96		144	STOP	193	PLAY
48	MKD$	97		145	PRINT	194	PSET
49		98		146	CLEAR	195	PRESET

Vervolg MSX-tokenlijst

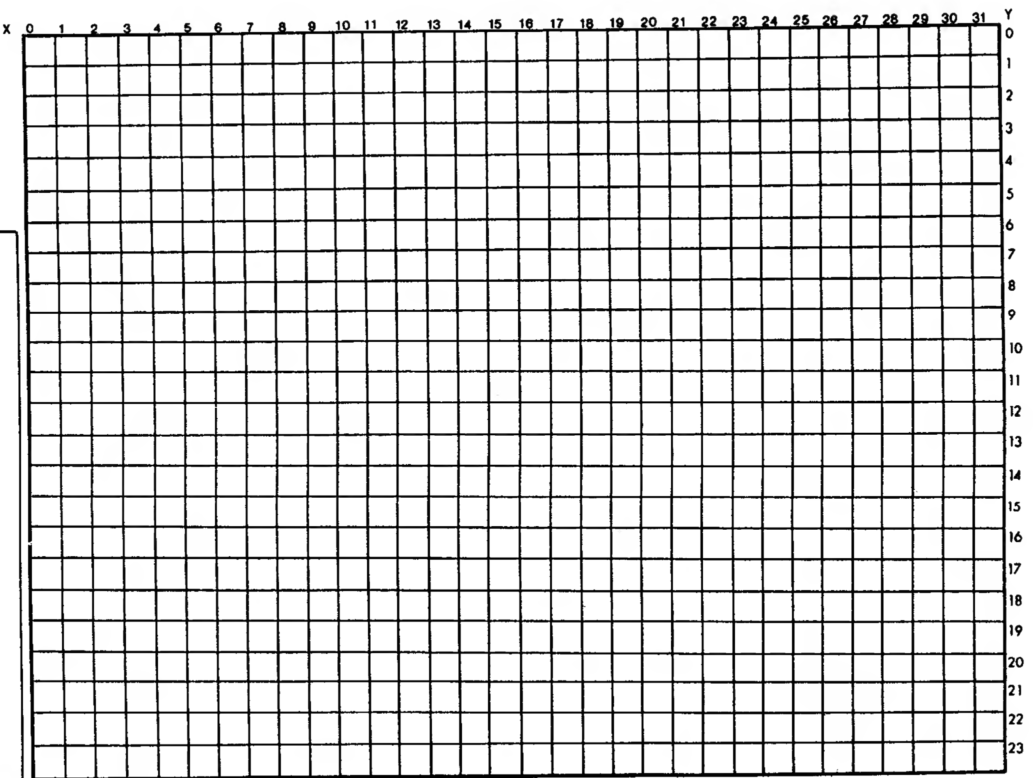
nr.	instruktie	nr.	instruktie
196	SOUND	227	STRING$
197	SCREEN	228	USING
198	VPOKE	229	INSTR
199	SPRITE$	230	´(ipv.REM)
200	VDP	231	VARPTR
201	BASE	232	CRLIN
202	CALL	233	ATTR$
203	TIME	234	DSKI$
204	KEY	235	OFF
205	MAX	236	INKEY$
206	MOTOR	237	POINT
207	BLOAD	238	>
208	BSAVE	239	=
209	DSKO$	240	<
210	SET	241	+
211	NAME	242	–
212	KILL	243	* (keer)
213	IPL	244	/
214	COPY	245	^
215	CMD	146	AND
216	LOCATE	247	OR
217	TO	248	XOR
218	THEN	249	EQV
219	TAB (	250	IMP
220	STEP	251	MOD
221	USR	252	\
222	FN (DEF	253	
223	SPC(	254	
224	NOT	255	
225	ERL	256	
226	ERR		

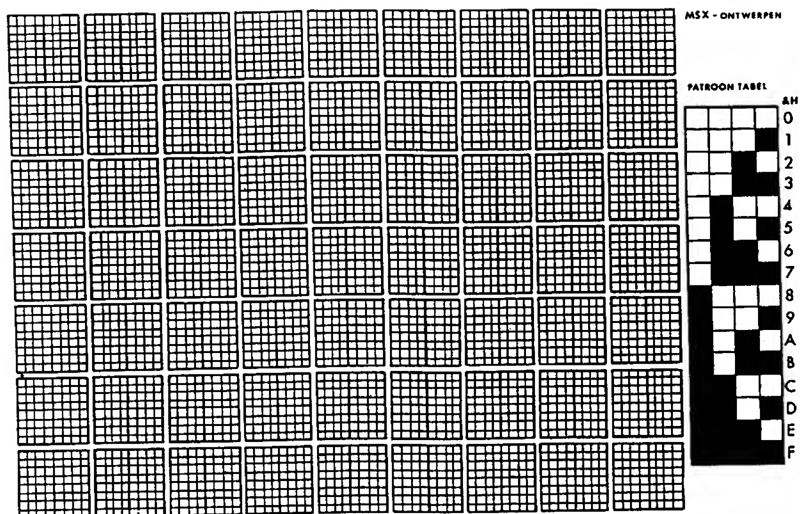
appendix 5
Ontwerpvellen

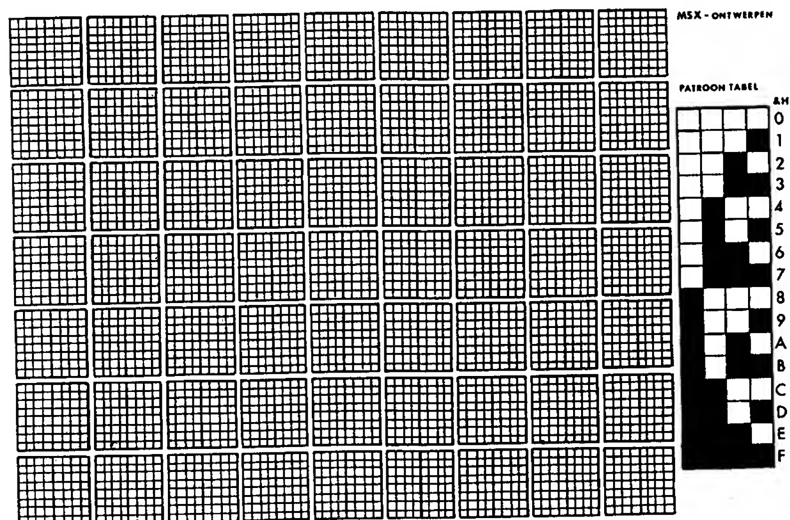
Tot de voorbereiding van een programma behoort de voorstelling op welke wijze het scherm zal worden ingedeeld. Aldus waar wat moet komen. Hiervoor treft u een schermontwerpvel aan met de verdeling 24x32. Voor screen1 kan dit worden gehandhaafd bij het gebruik van WIDTH 32. Screen1 leent zich verreweg het beste voor zowel tekst als grafische voorstellingen, zij het zonder de mogelijkheden van Screen2. Vanaf het voorgetekende schermontwerp zijn de locaties direkt af te lezen. Hierdoor zal het intypen van de regels voor het schermbeeld eenvoudiger worden. De schermposities zijn terug te vinden in de tabel in BASE(5).
U kunt fotokopieën van het ontwerpvel maken op bijvoorbeeld wat steviger papier. Wellicht zal er nog een ontwerpblok op de markt komen met onder andere zo'n ontwerpvel. Voor screen2 is dit reeds het geval. Het werken daarmee is lastig gezien de indeling van 256x 191 pickels. Bij het ontwerpvel hoort het matrix-vel. Het leent zich goed voor het ontwerpen van figuren voor sprites zowel als PRINT. Daarbij is de patroontabel met de hex.kodes direkt te raadplegen. Na enige tijd zal dit niet meer nodig zijn. Van het matrix-vel kunt u eveneens fotokopieën maken voor eigen gebruikt.
Er is iets voor te zeggen om uw ontwerpen te bewaren. Wat eens een geslaagd ontwerp was, zou naderhand nog eens van pas kunnen komen. Het gebruik van deze hulpmiddelen is een must. Het maakt het programmeren heel wat plezieriger.

MSX - SCREEN 1
schermposities
ontwerpvel

VPOKE / Base(5)
aantekeningen




MSX - ONTWERPEN
PATROON TABEL
&H
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F


MSX - ONTWERPEN
PATROON TABEL
&H
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

appendix 6
Listings

Opgenomen zijn alle listings van de programma's zoals ze in het boek worden gebruikt. Waar dit is aangegeven noodzaakt het tot het intikken van de listings. Omdat steeds aan de hand van een programma de tekst is opgezet. Bij de uitgever is eventueel een kassette of diskette met de programma's te bestellen. Opgemerkt moet worden dat de volgende listings exakt zijn weergegeven. Typefoutjes worden altijd gemaakt. Is een programma eenmaal ingetypt, zet deze dan eerst op kassette of floppy voordat u gaat runnen. Als er iets niet goed werkt of als u een foutmelding krijgt, dan is er een typefout gemaakt. Kontroleer de regels dan nog eens zorgvuldig. Liever de volgende dag dan meteen.

Programma-1: Bytes

```
10 REM ** schakelingen / CLOAD"BYTES" / progr.nr.1
 Basis van BASIC / Ca 4000 bytes
20 CLS:SCREEN2:COLOR 1,11,11:KEY OFF:WIDTH 32:OPEN
"GRP:"FOR OUTPUT AS 1
30 LINE(0,0)-(255,191),11,BF
40 GOSUB 870
50 K1=11:K2=11:K3=11:K4=11:K5=11:K6=11:K7=11:K8=11
:B1=0:BB=0
60 REM ** hoofdprogramma **
70 FOR I=0 TO 255
80 GOSUB 260
90 B=B+1:IF B=1 THEN 100 ELSE 110
100 F=11:G=6:GOTO 120
110 F=6:G=11:B=0
120 LINE(100,24)-(139,32),6,BF:PRESET(103,25),6:PR
INT#1,I
130 LINE(100,34)-(139,45),15,BF:A$=HEX$(I):PRESET(
115,37),15:PRINT#1,A$
140 LINE(100,46)-(139,55),5,BF:B$=OCT$(I):PRESET(1
11,48),5:PRINT#1,B$
150 LINE(57,79)-(182,87),12,BF:C$=BIN$(I):PRESET(9
1,80),12:PRINT#1,C$
160 FOR J=79 TO 56 STEP -2:FOR L=119 TO 120:PSET(L
```

```
,J),F:PSET(L-1,J-1),F:PSET(L-1,J),G:PSET(L,J-1),G:
NEXT J
170 FOR J=140 TO 188 STEP 2:PSET(J,29),F:PSET(J-1,
28),F:PSET(J-1,29),G:PSET(J,28),G:NEXT J
180 NEXT I
190 LINE(0,139)-(251,191),1,B:LINE(1,140)-(249,190
),3,BF
200 PRESET(8,143),3:PRINT#1,"Met deze 256 schakelv
arianten":PRESET(8,151),3:PRINT#1,"wordt alles ger
ealiseerd. De":PRESET(8,159),3:PRINT#1,"64-Kb MSX
bezit 81920 schakel-"
210 PRESET(8,167),3:PRINT#1,"eenheden; 16384 in VR
AM, 32768":PRESET(8,175),3:PRINT#1,"in ROM, 28815
in RAM en 3953":PRESET(8,183),3:PRINT#1,"in het sy
steemgebied"
220 GOTO 220
230 DEFUSR=&H69:A=USR(0)
240 END
250 GOTO 250
260 B1=B1+1:IF B1<=16 THEN 440 ELSE 270
270 B1=0:BB=BB+1
280 IF BB=1 THEN GOSUB 630
290 IF BB=2 THEN GOSUB 640
300 IF BB=3 THEN GOSUB 650
310 IF BB=4 THEN GOSUB 660
320 IF BB=5 THEN GOSUB 670
330 IF BB=6 THEN GOSUB 680
340 IF BB=7 THEN GOSUB 690
350 IF BB=8 THEN GOSUB 700
360 IF BB=9 THEN GOSUB 710
370 IF BB=10 THEN GOSUB 720
380 IF BB=11 THEN GOSUB 730
390 IF BB=12 THEN GOSUB 740
400 IF BB=13 THEN GOSUB 750
410 IF BB=14 THEN GOSUB 760
420 IF BB=15 THEN GOSUB 770
430 GOTO 260
440 ON B1 GOSUB 470,480,490,500,510,520,530,540,55
0,560,570,580,590,600,610,620
128 GOTO 780
460 REM 2e strip
470 RETURN
480 K8=1:RETURN
490 K7=1:K8=11:RETURN
500 K8=1:RETURN
510 K6=1:K7=11:K8=11:RETURN
```

```
520 K8=1:RETURN
530 K6=1:K7=1:K8=11:RETURN
540 K8=1:RETURN
550 K5=1:K6=11:K7=11:K8=11:RETURN
560 K8=1:RETURN
570 K7=1:K8=11:RETURN
580 K6=11:K8=1:RETURN
590 K6=1:K7=11:K8=11:RETURN
600 K6=1:K7=11:K8=1:RETURN
610 K7=1:K8=11:RETURN
620 K8=1:RETURN
630 K4=1:K5=11:K6=11:K7=11:K8=11:RETURN
640 K3=1:K4=11:K5=11:K6=11:K7=11:K8=11:RETURN
650 K3=1:K4=1:K5=11:K6=11:K7=11:K8=11:RETURN
660 K2=1:K3=11:K4=11:K5=11:K6=11:K7=11:K8=11:RETUR
N
670 K2=1:K3=11:K4=1:K5=11:K6=11:K7=11:K8=11:RETURN
680 K2=1:K3=1:K4=11:K5=11:K6=11:K7=11:K8=11:RETURN
690 K2=1:K3=1:K4=1:K5=11:K6=11:K7=11:K8=11:RETURN
700 K1=1:K2=11:K3=11:K4=11:K5=11:K6=11:K7=11:K8=11
:RETURN
710 K1=1:K2=11:K3=11:K4=1:K5=11:K6=11:K7=11:K8=11:
RETURN
720 K1=1:K2=11:K3=1:K4=11:K5=11:K6=11:K7=11:K8=11:
RETURN
730 K1=1:K2=11:K3=1:K4=1:K5=11:K6=11:K7=11:K8=11:R
ETURN
740 K1=1:K2=1:K3=11:K4=11:K5=11:K6=11:K7=11:K8=11:
RETURN
750 K1=1:K2=1:K3=11:K4=1:K5=11:K6=11:K7=11:K8=11:R
ETURN
760 K1=1:K2=1:K3=1:K4=11:K5=11:K6=11:K7=11:K8=11:R
ETURN
770 K1=1:K2=1:K3=1:K4=1:K5=11:K6=11:K7=11:K8=11:RE
TURN
780 LINE(58,105)-(69,117),K1,BF
790 LINE(74,105)-(85,117),K2,BF
800 LINE(90,105)-(101,117),K3,BF
810 LINE(106,105)-(117,117),K4,BF
820 LINE(122,105)-(133,117),K5,BF
830 LINE(138,105)-(149,117),K6,BF:BEEP
840 LINE(154,105)-(165,117),K7,BF:BEEP
850 LINE(170,105)-(179,117),K8,BF:BEEP
860 RETURN
870 REM ** schermbeeld **
880 LINE(55,102)-(182,120),1,B:LINE(56,103)-(181,1
```

```
19),1,B
890 FOR I=71 TO 182 STEP 16:LINE(I,103)-(I,119),1:
LINE(I+1,103)-(I+1,119),1:NEXT I
900 LINE(56,78)-(182,88),12,B:LINE(94,23)-(145,56)
,1,B:FOR L=64 TO 189 STEP 16:LINE(L,102)-(L,87),6:
NEXT L
910 PRESET(65,0):PRINT#1,"SCHAKELEENHEID":LINE(65,
8)-(173,8),4:PRESET(32,25):PRINT#1,"Decimaal":PRES
ET(0,37):PRINT#1,"Hexadecimaal":PRESET(48,48):PRIN
T#1,"Octaal"
920 PRESET(0,80):PRINT#1,"Binair:":PRESET(17,108):
PRINT#1,"BYTE:":PRESET(17,125):PRINT#1,"Bit :"
930 FOR I=54 TO 179 STEP 16:A=A+1:PRESET(I,125):PR
INT#1,AO I:PRESET(94,11):PRINT#1,"(adres)"
940 PRESET(208,0):PRINT#1,"CPU":LINE(186,7)-(252,1
23),6,B:LINE(188,9)-(250,121),6,BF
950 PRESET(192,11),6:PRINT#1,"Signaal":PRESET(192,
18),6:PRINT#1,"voor de":PRESET(192,26),6:PRINT#1,"
micro-":PRESET(192,34),6:PRINT#1,"proces-":PRESET(
192,42),6:PRINT#1,"sor-Z80"
960 PRESET(192,50),6:PRINT#1,"met de":PRESET(192,5
8),6:PRINT#1,"Clock-":PRESET(192,66),6:PRINT#1,"Pu
lse":PRESET(192,74),6:PRINT#1,"Genera-":PRESET(192
,82),6:PRINT#1,"tor(3½"
970 PRESET(192,90),6:PRINT#1,"miljoen":PRESET(192,
98),6:PRINT#1,"tikken":PRESET(192,106),6:PRINT#1,"
per se-":PRESET(192,114),6:PRINT#1,"conde)"
980 LINE(0,139)-(251,191),1,B:LINE(1,140)-(249,190
),3,BF
990 PRESET(8,143),3:PRINT#1,"VRAM:":PRESET(8,167),
3:PRINT#1,"ROM :":LINE(8,152)-(43,152),12:LINE(8,1
76)-(43,176),12
1000 PRESET(48,143),3:PRINT#1,"bytes representeren
 codes":PRESET(48,151),3:PRINT#1,"voor karakters,
kleuren":PRESET(48,159),3:PRINT#1,"en locaties"
1010 PRESET(48,167),3:PRINT#1,"bytes voor operator
s of":PRESET(48,175),3:PRINT#1,"deel operand; voor
waarde":PRESET(48,183),3:PRINT#1,"en/of adres deel
"
1020 RETURN
```

Programma-2: PEEK

Let op de te verwisselen REM-regels! Dit programma na het intikken in elk geval eerst wegschrijven op kassette of diskette.

```
10 REM ** PEEK-programma / CLOAD"PEEK"/ prog.nr.2
Basis van Basic / Ca 1300 bytes
20 CLS:SCREEN 1:COLOR 1,12:Y=5:X$=SPACE$(32):A=A-8
:G=0:N=0
30 P=1:A=A+8:FOR B=A TO A+7:C=VPEEK(B)
40 LOCATE 2,3:PRINT"Adres:";"      ";"Byte";"   ";"D
ec.";" ";"Kar.":LOCATE 1,4:PRINT STRING$(27,195)
50 REM P=0:A=A+8:FOR B=A TO A+7:C=PEEK(B)
60 REM LOCATE 2,3:PRINT"Adres:";"      ";"Byte";"
";"Dec.";" ";"Mnem.":LOCATE 1,4:PRINT STRING$(27,1
95)
70 A$=CHR$(C):B$=HEX$(C):D$=HEX$(B):Y=Y+1
80 LOCATE 0,Y:PRINT B
90 LOCATE 7,Y:PRINT D$
100 LOCATE 14,Y:PRINT B$
110 LOCATE 18,Y:PRINT C
120 IF P=1 THEN 220
130 IF C=0 THEN A$="NOP":GOTO 210
140 IF C=195 THEN A$="JP":GOTO 210
150 IF C=201 THEN A$="RET":GOTO 210
160 IF C=205 THEN A$="CALL":GOTO 210
170 IF C=211 THEN A$="OUT":GOTO 210
180 IF C=219 THEN A$="IN":GOTO 210
190 IF C=255 THEN A$="RST":GOTO 210
200 IF C>127 THEN 370
210 LOCATE 25,Y:PRINT A$:GOTO 370
220 IF B>7 AND B<256 THEN 320
230 IF B>256 AND B<2048 THEN 340
240 IF B>2048 AND B<4095 THEN 280
250 IF B>4096 AND B<6143 THEN 350
260 IF B>6143 AND B<6911 THEN 350
270 IF B>6911 AND B<16384 THEN 370
280 G=0:F=B-2048:IF F>7 AND F<256 THEN 320
290 IF F>256 AND F<2048 THEN 300 ELSE 370
300 N=N+1:IF N=1 THEN 310 ELSE 380
310 A$=CHR$(F/8):LOCATE 25,Y:PRINT A$:GOTO 370
320 N=N+1:IF N=1 THEN 330 ELSE 370
330 G=G+1:LOCATE 25,Y:PRINT CHR$(1);CHR$(G+64):GOT
O 370
340 N=N+1:IF N=1 THEN 360 ELSE 370
```

```
350 A$=CHR$(C):LOCATE 25,Y:PRINT A$:GOTO 370
360 A$=CHR$(B/8):LOCATE 25,Y:PRINT A$
370 NEXT B:N=0
380 PRINT:INPUT "Nieuw adres? Nee is RETURN";B:A=B
-8
390 FOR I=5 TO 17:LOCATE 0,I:PRINT X$:NEXT I:Y=5:I
F P=1 THEN 30 ELSE 50
```

Programma-3: LETTER

```
10 REM ** karakter wissel / CLOAD"LETTER" / prog.n
r.3 Basis van BASIC / Ca 4200 bytes
20 CLS:SCREEN1:COLOR 1,11,11:KEY OFF
30 CLS:PLAY"AFAF"
40 LOCATE 7,1:PRINT"HOOFD-MENU":LOCATE 0,3:PRINT"C
ontrole regel 23: TOETS 1":LOCATE 0,5:PRINT"LETTER
KLEUR: TOETS 2":LOCATE 0,7:PRINT"EINDE: TOETS 3"
50 A$=INKEY$:IF A$="" THEN 50
60 IF A$="1" THEN 100
70 IF A$="2" THEN 200
80 IF A$="3" THEN 1210
90 GOTO 50
100 CLS:SCREEN1:PLAY"AFAF"
110 LOCATE10,1:PRINT"MENU":LOCATE0,3:PRINT"SCREEN0
: TOETS 1":LOCATE0,4:PRINT"SCREEN1: TOETS 2":LOCAT
E0,5:PRINT"SCREEN2: TOETS 3":LOCATE0,7:PRINT"VPOKE
 SCREEN0: TOETS 4":LOCATE0,8:PRINT"VPOKE SCREEN1:
TOETS 5":LOCATE0,10:PRINT"TERUG HOOFDMENU: TOETS 6
"
120 B$=INKEY$:IF B$="" THEN 120
130 IF B$="1" THEN GOSUB 1100
140 IF B$="2" THEN GOSUB 1110
150 IF B$="3" THEN GOSUB 1160
160 IF B$="4" THEN GOSUB 1030
170 IF B$="5" THEN GOSUB 950
180 IF B$="6" THEN 30
190 FOR D=1 TO 750:NEXT D:GOTO 100
200 CLS:SCREEN1:COLOR1,11,11
210 CLS:PLAY"AFAF"
220 LOCATE10,1:PRINT"MENU":LOCATE0,3:PRINT"Letterk
leur: TOETS 1":LOCATE0,5:PRINT"TERUG HOOFDMENU: TO
ETS 2":LOCATE0,8:PRINT"OVERIGE TOETSEN OP HET SCHE
RMVERMELD"
230 X$=INKEY$:IF X$="" THEN 230
240 IF X$="1" THEN GOSUB 270
```

```
250 IF X$="2" THEN 30
260 GOTO 230
270 CLS:COLOR 12,1,1
280 GOSUB 380:GOSUB 540
290 LOCATE 0,18:PRINT"Geel-rood:  TOETS 1":LOCATEO
,19:PRINT"Blauw-geel: TOETS 2":LOCATEO,20:PRINT"Wi
t-blauw: TOETS 3":LOCATEO,21:PRINT"Zwart-groen: TO
ETS 4":LOCATEO,22:PRINT"Input tekst/kleur: TOETS 5
"
300 Y$=INKEY$:IF Y$="" THEN 300
310 IF Y$="1" THEN GOSUB 680
320 IF Y$="2" THEN GOSUB 710
330 IF Y$="3" THEN GOSUB 740
340 IF Y$="4" THEN GOSUB 770
350 IF Y$="5" THEN GOSUB 800
360 IF Y$="6" THEN 30
370 GOTO 300
380 REM ** kleur vpoken **
390 A=BASE(6)
400 FOR C=0 TO 7:VPOKE A+(C/8),&H66:NEXT C:FOR C=8
 TO 15:VPOKE A+(C/8),&HAA:NEXT C:FOR C=16 TO 23:VP
OKE A+(C/8),&H44:NEXT C:FOR C=24 TO 31:VPOKE A+(C/
8),&HCC:NEXT C:FOR C=128 TO 135:VPOKE A+(C/8),&HDD
:NEXT C
410 FOR J=2 TO 5:FOR I=10 TO 17:LOCATE I,J:PRINT"Ç
":NEXT I,J
420 FOR C=136 TO 151:VPOKE A+(C/8),&HA6:NEXT C:FOR
 C=152 TO 167:VPOKE A+(C/8),&H4A:NEXT C:FOR C=168
TO 183:VPOKE A+(C/8),&HF4:NEXT C:FOR C=184 TO 199:
VPOKE A+(C/8),&HC:NEXT C
430 REM ** letterwissel **
440 A=BASE(7)+520:B=BASE(7)+1088:FOR I=0 TO 7:C=VP
EEK(A+I):VPOKE B+I,C:NEXT I
450 D=BASE(7)+600:B=BASE(7)+1096:FOR I=0 TO 7:C=VP
EEK(D+I):VPOKE B+I,C:NEXT I
460 E=BASE(7)+656:B=BASE(7)+1104:FOR I=0 TO 23:C=V
PEEK(E+I):VPOKE B+I,C:NEXT I
470 F=BASE(7)+800:B=BASE(7)+1128:FOR I=0 TO 15:C=V
PEEK(F+I):VPOKE B+I,C:NEXT I
480 G=BASE(7)+840:B=BASE(7)+1144:FOR I=0 TO 7:C=VP
EEK(G+I):VPOKE B+I,C:NEXT I
490 H=BASE(7)+824:B=BASE(7)+1152:FOR I=0 TO 7:C=VP
EEK(H+I):VPOKE B+I,C:NEXT I
500 K=BASE(7)+880:B=BASE(7)+1160:FOR I=0 TO 7:C=VP
EEK(K+I):VPOKE B+I,C:NEXT I
510 RETURN
```

```
520 LOCATE 9,0:PRINT"LETTERKLEUR":LOCATE 8,18:PRIN
T"STARKe dingen":LOCATE 8,19
530 LOCATE 4,8:PRINT"1":LOCATE 24,8:PRINT"2":LOCAT
E 4,13:PRINT"3":LOCATE 24,13:PRINT"4"
540 LOCATE 11,3:PRINT"STARKe":LOCATE 11,4:PRINT"di
ngen"
550 REM ** kleurkaders **
560 FOR J=7 TO 10:FOR I=6 TO 13:LOCATE I,J:PRINT"
A":NEXT I,J
570 FOR J=7 TO 10:FOR I=15 TO 22:LOCATE I,J:PRINT"
█":NEXT I,J
580 FOR J=12 TO 15:FOR I=6 TO 13:LOCATE I,J:PRINT"
┼":NEXT I,J
590 FOR J=12 TO 15:FOR I=15 TO 22:LOCATE I,J:PRINT
"┌":NEXT I,J
600 RETURN
610 REM ** karakterkleur **
620 A=BASE(6)
630 FOR C=136 TO 151:VPOKE A+(C/8),K:NEXT C
640 RETURN
650 REM ** kleurtekst **
660 LOCATE X,Y:PRINT"ïîêèëÄ":LOCATE X,Y+1:PRINT"ìA
æÉÄæ"
670 RETURN
680 REM ** geel op rood **
690 GOSUB 550:K=166:GOSUB 610:X=7:Y=8:GOSUB 650
700 RETURN
710 REM ** blauw op geel **
720 GOSUB 550:K=74:GOSUB 610:X=16:Y=8:GOSUB 650
730 RETURN
740 REM ** wit op blauw **
750 GOSUB 550:K=244:GOSUB 610:X=7:Y=13:GOSUB 650
760 RETURN
770 REM ** zwart op groen **
780 GOSUB 550:K=28:GOSUB 610::X=16:Y=13:GOSUB 650
790 RETURN
800 REM ** input tekst **
810 CLS:GOSUB 420:GOSUB 820:GOTO 880
820 LOCATE 1,1:PRINT"S=(139)":LOCATE 1,2:PRINT"T=(
140)":LOCATE 1,3:PRINT"A=(136)":LOCATE 1,4:PRINT"R
=(138)":LOCATE 1,5:PRINT"K=(137)"
830 LOCATE 1,6:PRINT"e=(142)":LOCATE 1,7:PRINT"d=(
141)":LOCATE 1,8:PRINT"i=(143)":LOCATE 1,9:PRINT"n
=(145)":LOCATE 1,10:PRINT"g=(144)":LOCATE 1,11:PRI
NT"e=(142)":LOCATE 1,12:PRINT"n=(145)"
```

```
840 LOCATE 10,0:PRINT"ïîêèëÄ ìÄæÉÄæ":LOCATE 10,1:P
RINT"STARKe dingen"
850 LOCATE 10,3:PRINT"ëÄæì Äæ ìÄ èÄæÉ":LOCATE 10,4
:PRINT"Kind in de Ring"
860 LOCATE 10,6:PRINT"îÄÉÄæ ÄÄæ ëèêîï":LOCATE 10,7
:PRINT"Tegen een KRATS":LOCATE 10,9:PRINT"èÄÉÄæ Äæ
 ÉÄìÄæÉ":LOCATE 10,10:PRINT"Regen in geding":LOCAT
E10,12:PRINT"ÉèêêÉ ÉÄìêêæ!":LOCATE10,13:PRINT"gRAA
g gedAAn!"
870 RETURN
880 LOCATE 0,19:INPUT"Welke letterkleur(gebruik:33
-49-65-81-97-113-129-145-161-177-193-209-225-241 O
F 256 voor return menu) ";K:IF K>255 THEN 30 ELSE
GOSUB 610
890 CLS:GOSUB 820
900 LOCATE 0,19:INPUT"Tekst(Max 28 tekens: gebruik
 vermelde karaktercodes of RE-TURN); terug menu=@
";B$
910 IF B$="@" THEN 30
920 IF LEN(B$)>28 THEN 930
930 CLS:GOSUB 820:GOTO 880
940 REM ** regel 23 **
950 CLS:SCREEN1:COLOR1,10,10:A=64
960 B=BASE(6)
970 FOR C=1 TO 31:VPOKE B+(C/8),&H1A:NEXT C
980 D=BASE(5)+736
990 FOR I=0 TO 32
1000 A=A+1
1010 VPOKE D+I,A
1020 NEXT I:RETURN
1030 CLS:SCREEN0:COLOR 1,11,11:A=64
1040 FOR C=1 TO 31:VPOKE B+(C/8),&H1A:NEXT C
1050 D=BASE(0)+920
1060 FOR I=0 TO 39
1070 A=A+1
1080 VPOKE D+I,A
1090 NEXT I:RETURN
1100 CLS:SCREEN0:COLOR4,10,1:A=-1:GOTO 1120
1110 CLS:SCREEN1:COLOR4,10,1:A=-1
1120 FOR I=0 TO 23
1130 A=A+1
1140 LOCATE 1,I:PRINT"REGEL: ";A
1150 NEXT I:RETURN
1160 CLS:SCREEN2:COLOR 4,10,1:OPEN"GRP:"FOR OUTPUT
 AS 1:A=-1
1170 FOR I=0 TO 191 STEP 8
```

```
1180 A=A+1
1190 PRESET(7,I),15:PRINT#1,"REGEL: ";A
1200 NEXT I:RETURN
1210 CLS:END
```

Programma-4: COLOR3

```
10 REM ** COLOR 3 PROGRAMMA / CLOAD"COLOR3" / prog
r.nr.4 Basis van BASIC / Ca 6800 bytes / na RUN ca
 20000 bytes!
20 CLS:SCREEN3:OPEN"GRP:"FOR OUTPUT AS 1:Q=-1:DIM
F(500,2)
30 DEFUSR=&H69:A=USR(0)
40 W$=INKEY$:IF W$="" THEN 30
50 IF W$="1" THEN GOSUB 120
60 IF W$="2" THEN 230
70 IF W$="3" THEN GOSUB 1030
80 IF W$="4" THEN GOSUB 1110
90 IF W$="5" THEN GOSUB 1190
100 IF W$="6" THEN 1690
110 GOTO 30
120 REM ** grafische instructies **
130 CLS:SCREEN3,1:COLOR 1,15,15:RESTORE 190
140 LINE(0,0)-(255,191),15,BF:LINE(0,159)-(255,191
),6,B
150 PRESET(25,50),4:DRAW"D40R60U40L20U20L20D20L20"
160 CIRCLE(150,66),64,12,,,1.4:PAINT(150,66),12:PS
ET(56,15),13
170 PRESET(44,116):PRINT#1,"3"
180 F=BASE(14):FOR I=0 TO 7:READ Z$:VPOKE F+I,VAL(
"&H"+Z$):NEXT I
190 DATA 00,0F,1E,3C,F8,3C,1E,0F,00
200 FOR I=1500 TO 0 STEP -1:PUT SPRITE 0,(I,52),10
,0:NEXT I
210 PLAY"AFAF"
220 RETURN
230 REM ** kleuren **
240 SCREEN2:COLOR 1,15,15:C=0:A=0
250 A$="transperant":B$="zwart":C$="medium-groen":
D$="licht-groen":E$="donker-blauw":F$="licht-blauw
":G$="donker-rood":H$="cyaan":I$="medium-rood":J$=
"licht-rood":K$="donker-geel":L$="licht-geel":M$="
donker-groen":N$="magenta":O$="grijs":P$="wit"
260 RESTORE 310:F=BASE(14):VDP(1)=227
270 FOR I=0 TO 31
```

```
280 READ Z$
290 VPOKE F+I,VAL("&H"+Z$)
300 NEXT I
310 DATA 1F,0F,00,00,00,00,01,00,00,01,00,00,00,00
,0F,1F,F8,FC,06,06,06,06,86,FC,F8,84,06,06,06,06,F
C,F8
320 LINE(0,0)-(255,191),15,BF
330 PRESET(23,42):PRINT#1,"Kleurcode:":PRESET(127,
42):PRINT#1,"Kleurcode:":PRESET(47,63):PRINT#1,"Kl
eurcombinatie:"
340 PRESET(22,86):PRINT#1,"SCREEN":PRESET(15,27):P
RINT#1,"A":PRESET(228,27):PRINT#1,"B":LINE(17,82)-
(73,94),1,B
350 PRESET(7,136):PRINT#1,"CODE:":LINE(7,144)-(40,
144),1::PRESET(7,148):PRINT#1,"Dec.:":PRESET(7,160
):PRINT#1,"Hex.:":PRESET(7,172):PRINT#1,"Oct.:":PR
ESET(7,184):PRINT#1,"Bin.:"
360 LINE(7,2)-(239,79),4,B:LINE(5,0)-(241,81),6,B:
LINE(123,2)-(123,55),1:LINE(87,56)-(159,56),4
370 FOR I=112 TO 168 STEP 8:A=A+1:PRESET(71,I):PRI
NT#1,A:NEXT I
380 LINE(87,111)-(135,175),4,B:LINE(111,111)-(111,
175),4
390 FOR I=119 TO 167 STEP 8:LINE(88,I)-(135,I),4:N
EXT I:PRESET(95,103):PRINT#1,"A  B":PRESET(143,88)
:PRINT#1,"Schermopbouw:":LINE(143,96)-(239,96),1
400 PRESET(143,104):PRINT#1,"32x8 blokjes":PRESET(
143,114):PRINT#1,"per regel":PRESET(143,134):PRINT
#1,"6 regels":PRESET(143,154):PRINT#1,"256 kleur-"
:PRESET(143,164):PRINT#1,"blokjes per"
410 PRESET(143,174):PRINT#1,"regel en 1536":PRESET
(143,184):PRINT#1,"op het scherm"
420 FOR K=0 TO 15
430 GOSUB 630
440 IF K=0 OR K=15 THEN 450 ELSE 470
450 LINE(23,23)-(119,39),15,BF
460 LINE(23,23)-(119,39),1,B:GOTO 480
470 LINE(23,23)-(119,39),K,BF
480 FOR V=0 TO 15:C=C+1
490 IF C=255 OR V=0 OR V=15 THEN CL=13 ELSE CL=V
500 PUT SPRITE 0,(8+V*2.5,98),CL,0
510 Q$=HEX$(C-1):R$=OCT$(C-1):S$=BIN$(C-1)
520 LINE(176,63)-(199,71),15,BF:PRESET(176,63):PRI
NT#1,USING"###";C-1
530 LINE(44,95)-(74,182),15,BF:PRESET(44,148):PRIN
```

```
T#1,C-1
540 LINE(52,180)-(140,191),15,BF:PRESET(52,160):PR
INT#1,Q$:PRESET(52,172):PRINT#1,R$:PRESET(52,184):
PRINT#1,S$
550 IF V=0 OR V=15 THEN 560 ELSE 580
560 LINE(127,23)-(226,39),15,BF
570 LINE(127,23)-(222,39),1,B:GOTO 590
580 LINE(127,23)-(222,39),V,BF
590 GOSUB 830
600 NEXT V:NEXT K
610 PLAY"AFAF"
620 GOTO 30
630 ON K+1 GOSUB 670,680,690,700,710,720,730,740,7
50,760,770,780,790,800,810,820
640 LINE(23,7)-(119,15),15,BF:PRESET(X,8):PRINT#1,
X$
650 LINE(103,40)-(119,48),15,BF:PRESET(105,42):PRI
NT#1,USING"##";K
660 FOR D=1 TO 350:NEXT D:RETURN
670 X$=A$:X=31:RETURN
680 X$=B$:X=79:RETURN
690 X$=C$:X=23:RETURN
700 X$=D$:X=31:RETURN
710 X$=E$:X=23:RETURN
720 X$=F$:X=31:RETURN
730 X$=G$:X=31:RETURN
740 X$=H$:X=79:RETURN
750 X$=I$:X=31:RETURN
760 X$=J$:X=39:RETURN
770 X$=K$:X=31:RETURN
780 X$=L$:X=39:RETURN
790 X$=M$:X=23:RETURN
800 X$=N$:X=63:RETURN
810 X$=O$:X=79:RETURN
820 X$=P$:X=95:RETURN
830 ON V+1 GOSUB 870,880,890,900,910,920,930,940,9
50,960,970,980,990,1000,1010,1020
840 LINE(127,7)-(222,15),15,BF:PRESET(127,8):PRINT
#1,Y$
850 LINE(209,40)-(223,48),15,BF:PRESET(209,42):PRI
NT#1,USING"##";V
860 FOR D=1 TO 350:NEXT D:RETURN
870 Y$=A$:RETURN
880 Y$=B$:RETURN
890 Y$=C$:RETURN
900 Y$=D$:RETURN
```

```
910 Y$=E$:RETURN
920 Y$=F$:RETURN
930 Y$=G$:RETURN
940 Y$=H$:RETURN
950 Y$=I$:RETURN
960 Y$=J$:RETURN
970 Y$=K$:RETURN
980 Y$=L$:RETURN
990 Y$=M$:RETURN
1000 Y$=N$:RETURN
1010 Y$=O$:RETURN
1020 Y$=P$:RETURN
1030 REM ** kleurbeeld 1 **
1040 CLS:SCREEN3:COLOR 1,1,1:LINE(0,0)-(255,191),1
,BF
1050 FOR I=0 TO 1536
1060 C=RND(1)*255
1070 VPOKE A+I,C
1080 NEXT I
1090 PLAY"AFAF"
1100 RETURN
1110 REM ** kleurbeeld 2 **
1120 CLS:SCREEN3:COLOR 15,1,1:LINE(0,0)-(255,191),
1,BF
1130 FOR K=40 TO 202 STEP 4
1140 FOR R=40 TO 180 STEP 4
1150 PSET(K,R),((R/6.25+COS((K-40)/80*3.14159)*8+8
)MOD 14+1)
1160 NEXT R,K
1170 PLAY"AFAF"
1180 RETURN
1190 REM ** DATA kleuren **
1200 CLS:COLOR 15,5,5:LINE(0,0)-(255,191),5,BF
1210 E=BASE(17)+304:RESTORE 1470:N=0
1220 FOR I=0 TO 7
1230 READ KLEUR
1240 VPOKE E,KLEUR:E=E+1
1250 NEXT I
1260 N=N+1:IF N<19 THEN 1220
1270 IF N=19 THEN E=560
1280 IF N<38 THEN 1220
1290 IF N=38 THEN E=816
1300 IF N<57 THEN 1220
1310 IF N=57 THEN E=1072
1320 IF N<76 THEN 1220
```

```
1330 I=86:J=2
1340 PRESET(I,J):PRINT #1,"M"
1350 PRESET(I+24,J):PRINT#1,"S"
1360 PRESET(I+48,J):PRINT#1,"X"
1370 I=32:J=167
1380 PRESET(I,J):PRINT#1,"S"
1390 PRESET(I+24,J):PRINT#1,"C"
1400 PRESET(I+48,J):PRINT#1,"R"
1410 PRESET(I+72,J):PRINT#1,"E"
1420 PRESET(I+96,J):PRINT#1,"E"
1430 PRESET(I+120,J):PRINT#1,"N"
1440 PRESET(I+166,J):PRINT#1,"3"
1450 PLAY"AFAF"
1460 RETURN
1470 DATA 170,172,172,164,172,172,164,169
1480 DATA 170,196,73,153,73,196,204,76,238,204,76,
148,76,204,196,73
1490 DATA 170,76,204,196,204,76,148,153,238,196,73
,153,73,196,204,76,238,204,76,148,76,204,196,73
1500 DATA 170,76,204,196,204,76,148,153,238,196,73
,153,73,196,204,76,238,204,76,148,76,204,196,73
1510 DATA 170,76,204,196,204,76,148,153,238,196,73
,153,73,196,204,76,170,204,76,148,76,204,196,73
1520 DATA 170,76,204,196,204,76,148,153,238,196,73
,153,73,196,204,76,170,204,76,148,76,204,196,73
1530 DATA 170,76,204,196,204,76,148,153,238,196,73
,153,73,196,204,76,170,204,76,148,76,204,196,73,16
1,161,161,161,161,161,161,161
1540 REM LAAG 2
1550 DATA 164,172,172,164,172,172,164,169,204,196,
73,153,73,196,204,76,196,204,76,148,76,204,196,73,
148,76,204,196,204,76,148,153,204,196,73,153,73,19
6,204,76
1560 DATA 196,204,76,148,76,204,196,73,148,76,17,2
2,22,22,31,31,204,196,17,102,102,102,255,255,196,2
04,17,102,102,102,255,255,148,76,17,102,102,102,25
5,255
1570 DATA 204,196,17,102,102,102,255,255,196,204,1
7,102,102,102,255,255,148,76,28,20,28,28,20,25,204
,196,73,153,73,196,204,76,196,204,76,148,76,204,19
6,73
1580 DATA 148,76,204,196,204,76,148,153,204,196,73
,153,73,196,204,76,196,204,76,148,76,204,196,73,22
5,225,225,225,225,225,225,225
1590 REM LAAG 3
```

```
1600 DATA 164,172,172,164,172,172,164,169,204,196,
73,153,73,196,204,76,196,204,76,148,76,204,196,73,
148,76,204,196,204,76,148,153,204,196,73,153,73,19
6,204,76
1610 DATA 196,204,76,148,76,204,196,73,31,20,20,20
,17,76,148,153,255,68,68,68,17,196,204,76,255,68,6
8,68,17,204,196,73,255,68,68,68,17,76,148,153
1620 DATA 255,68,68,68,17,196,204,76,255,68,68,68,
17,204,196,73,20,28,28,20,28,76,148,153,204,196,73
,153,73,196,204,76,196,204,76,148,76,204,196,73
1630 DATA 148,76,204,196,204,76,148,153,204,196,73
,153,73,196,204,76,196,204,76,148,76,204,196,73,16
1,161,161,161,161,161,161,161
1640 REM LAAG 4
1650 DATA 164,172,172,164,172,172,170,17,204,196,7
3,153,73,196,170,17,196,204,76,148,76,204,170,17,1
48,76,204,196,204,76,170,17,204,196,73,153,73,196,
170,17
1660 DATA 196,204,76,148,76,204,170,17,148,76,204,
196,204,76,170,17,204,196,73,153,73,196,170,17,196
,204,76,148,76,204,170,17,148,76,204,196,204,76,17
0,17
1670 DATA 204,196,73,153,73,196,170,17,196,204,76,
148,76,204,170,17,148,76,204,196,204,76,170,17,204
,196,73,153,73,196,170,17,196,204,76,148,76,204,17
0,17
1680 DATA 148,76,204,196,204,76,170,17,204,196,73,
153,73,196,170,17,196,204,76,148,76,204,170,17,161
,161,161,161,161,161,161,17
1690 CLS:END
```

Programma-5: CALCUL

```
10 REM SCHERMLEZING / CLOAD"CALCUL" / progr.nr.5 B
asis van BASIC / Ca 8100 bytes
20 CLS:SCREEN1,0:KEY OFF:COLOR 1,14,14:WIDTH 32
30 LOCATE12,2:PRINT"MENU":LOCATE 2,4:PRINT"TOETS 1
: schermlezing":LOCATE2,6:PRINT"TOETS 2: calculato
r":LOCATE2,8:PRINT"TOETS 3: einde programma"
40 A$=INKEY$:IF A$="" THEN 40
50 IF A$="1" THEN GOSUB 240
60 IF A$="2" THEN 100
70 IF A$="3" THEN 2130
80 DEFUSR=&H69:A=USR(0)
90 CLS:PLAY"AFAF":GOTO 30
100 REM ** aanwijzen **
```

```
110 CLS:X=140:Y=138:GOSUB 1470
120 F=STICK(0)
130 IF F=1 THEN Y=Y-1.5
140 IF F=2 THEN X=X+1.5:Y=Y-1.5
150 IF F=3 THEN X=X+1.5
160 IF F=4 THEN X=X+1.5:Y=Y+1.5
170 IF F=5 THEN Y=Y+1.5
180 IF F=6 THEN X=X-1.5:Y=Y+1.5
190 IF F=7 THEN X=X-1.5
200 IF F=8 THEN X=X-1.5:Y=Y-1.5
210 PUT SPRITE 0,(X,Y),1,0
220 IF STRIG(0) THEN GOSUB 380
230 GOTO 120
240 REM ** codes lezen **
250 CLS:GOSUB 2060
260 LOCATE 5,5:PRINT"CODES":LOCATE 5,7:PRINT"VIA":
LOCATE5,9:PRINT"VPEEK":LOCATE5,11:PRINT"VAN HET SC
HERM":LOCATE 5,13:PRINT"GELEZEN":LOCATE5,15:PRINT"
IN BASE(5)"
270 LOCATE5,17:PRINT"☺ ☻ ♥ ♦ ♣ ♪ ♫ ☼":LOCA
TE5,19:PRINT"STOP-toets voor controle"
280 A=BASE(5):P=-1
290 FOR J=0 TO 22
300 FOR I=0 TO 31
310 P=P+1
320 PUT SPRITE 1,(P*8,INT(P/32)*8-2),13,0
330 C=VPEEK(A+P)
340 LOCATE0,22:PRINT"Schermpositie=";P;"CODE=";C
350 FOR D=1 TO 30:NEXT D
360 NEXT I,J
370 RETURN
380 REM ** calculator **
390 K=INT(X/8):R=INT(Y/8)
400 A=BASE(5)
410 C=VPEEK(A+(R*32+K))
420 IF K<11 OR K>18 THEN 720
430 IF R<5 OR R>18 THEN 720
440 IF C=8 THEN A$="F":GOSUB 810
450 IF C=12 THEN GOSUB 740
460 IF C=128 THEN A$="1":GOSUB 810
470 IF C=132 THEN A$="2":GOSUB 810
480 IF C=136 THEN A$="3":GOSUB 810
490 IF C=140 THEN A$="4":GOSUB 810
500 IF C=144 THEN A$="5":GOSUB 810
510 IF C=148 THEN A$="6":GOSUB 810
520 IF C=152 THEN A$="7":GOSUB 810
```

```
530 IF C=156 THEN A$="8":GOSUB 810
540 IF C=160 THEN A$="9":GOSUB 810
550 IF C=164 THEN A$="0":GOSUB 810
560 IF C=168 THEN N=1
570 IF C=172 THEN N=2
580 IF C=176 THEN N=3
590 IF C=180 THEN N=4
600 IF C=184 THEN N=5
610 IF C=188 THEN N=6
620 IF C=208 THEN N=7
630 IF C=212 THEN N=8
640 IF C=216 THEN N=9
650 IF C=220 THEN GOSUB 1250
660 IF C=224 THEN H=1
670 IF C=228 THEN A$="A":GOSUB 810
680 IF C=232 THEN A$="B":GOSUB 810
690 IF C=236 THEN A$="C":GOSUB 810
700 IF C=240 THEN A$="D":GOSUB 810
710 IF C=244 THEN A$="E":GOSUB 810
720 RETURN
730 REM ** verwerkingroutines **
740 Z$=SPACE$(12):LOCATE9,2:PRINT Z$
750 X$=SPACE$(32):Y$=SPACE$(30):LOCATE0,21:PRINT X
$:LOCATE0,22:PRINT Y$
760 A$="":H$="":G=0:H=0
770 I$(1)="":I$(2)="":I$(3)="":I$(4)=""
780 B$(1)="":B$(2)="":B$(3)="":B$(4)="":H$=""
790 RETURN
800 LOCATE10,2:PRINT"TE GROOT!":PLAY"CACA":FOR D=1
 TO 500:NEXT D:GOSUB 740:RETURN
810 IF H=0 THEN 1240
820 G=G+1:IF G>4 THEN 800
830 B$(G)=A$:H$=B$(1)+B$(2)+B$(3)+B$(4)
840 IF G<=4 THEN 850 ELSE IF G=4 THEN 860
850 LOCATE 12,2:PRINT"&H";H$:GOTO 1240
860 IF LEN(H$)<4 THEN 870 ELSE 880
870 A=0:B=4095:GOTO 1190
880 C$=MID$(H$,1,1)
890 IF C$="1" THEN 1040
900 IF C$="2" THEN 1050
910 IF C$="3" THEN 1060
920 IF C$="4" THEN 1070
930 IF C$="5" THEN 1080
940 IF C$="6" THEN 1090
950 IF C$="7" THEN 1100
960 IF C$="8" THEN 1110
```

```
970 IF C$="9" THEN 1120
980 IF C$="A" THEN 1130
990 IF C$="B" THEN 1140
1000 IF C$="C" THEN 1150
1010 IF C$="D" THEN 1160
1020 IF C$="E" THEN 1170
1030 IF C$="F" THEN 1180 ELSE 1240
1040 A=4096:B=8191:GOTO 1190
1050 A=8192:B=12287:GOTO 1190
1060 A=12288:B=16383:GOTO 1190
1070 A=16384:B=20479:GOTO 1190
1080 A=20480:B=24575:GOTO 1190
1090 A=24576:B=28671:GOTO 1190
1100 A=28672:B=32767:GOTO 1190
1110 A=32768!:B=36863!:GOTO 1190
1120 A=36864!:B=40959!:GOTO 1190
1130 A=40960!:B=45055!:GOTO 1190
1140 A=45056!:B=49151!:GOTO 1190
1150 A=49152!:B=53247!:GOTO 1190
1160 A=53248!:B=57343!:GOTO 1190
1170 A=57344!:B=61439!:GOTO 1190
1180 A=61440!:B=65535!
1190 FOR I=A TO B
1200 HX$=HEX$(I)
1210 IF HX$=H$ THEN 1230
1220 NEXT I
1230 PLAY"AFAF":LOCATE9,2:PRINT"Dec.:";I
1240 RETURN
1250 IF H=1 THEN 860
1260 X$=SPACE$(32):Y$=SPACE$(30):LOCATE0,21:PRINT
X$:LOCATE0,22:PRINT Y$
1270 ON N GOSUB 1290,1310,1330,1350,1370,1390,1410
,1430,1450
1280 RETURN
1290 LOCATE0,21:PRINT"Opgeteld staat netjes zul je
 be-doelen!"
1300 RETURN
1310 LOCATE0,21:PRINT"Wat zie jij er afgetrokken u
it!"
1320 RETURN
1330 LOCATE4,21:PRINT"Vermenigvuldig je zelf!"
1340 RETURN
1350 LOCATE0,21:PRINT"Samen alles eerlijk delen...
ja, ja!"
1360 RETURN
```

```
1370 LOCATE0,21:PRINT"Wortels plus uien...ook zo g
ek  op hutspot?"
1380 RETURN
1390 LOCATE0,21:PRINT"De machtsverhoudingen lijken
 nognergens naar..."
1400 RETURN
1410 LOCATE0,21:PRINT"CHR$? Zo ongeletterd zul je
nietwezen..."
1420 RETURN
1430 LOCATE0,21:PRINT"Hexenjacht is iets voor spoo
k-  rijders..."
1440 RETURN
1450 LOCATE2,21:PRINT"Pi verkoopt uilen in Athene"
1460 RETURN
1470 REM ** schermbeeld **
1480 A=BASE(6)
1490 FOR C=8 TO 15:VPOKE A+(C/8),&H6B:NEXT C:FOR C
=16 TO 31:VPOKE A+(C/8),&HB8:NEXT C:FOR C=208 TO 2
47:VPOKE A+(C/8),&H6B:NEXT C
1500 FOR C=128 TO 191:VPOKE A+(C/8),&H6B:NEXT C:FO
R C=192 TO 199:VPOKE A+(C/8),&HDE:NEXT C:FOR C=200
 TO 207:VPOKE A+(C/8),&H88:NEXT C
1510 A=BASE(7)+1024:RESTORE 1630
1520 FOR I=0 TO 639:READ C$
1530 VPOKE A+I,VAL("&H"+C$)
1540 NEXT I
1550 A=BASE(7)+1664:RESTORE 1840
1560 FOR I=0 TO 319:READ C$
1570 VPOKE A+I,VAL("&H"+C$)
1580 NEXT I
1590 A=BASE(7)+64
1600 FOR I=0 TO 63:READ C$
1610 VPOKE A+I,VAL("&H"+C$)
1620 NEXT I
1630 REM DATA-regels calcul.**
1640 DATA FF,80,80,81,83,81,81,81,81,81,81,80,80,F
F,80,40,FE,02,02,02,02,02,02,02,02,02,02,02,02,FE,
02,01
1650 DATA FF,80,80,83,84,84,80,80,81,82,87,80,80,F
F,80,40,FE,02,02,82,42,42,42,82,02,02,C2,02,02,FE,
02,01
1660 DATA FF,80,80,83,84,80,81,80,80,84,83,80,80,F
F,80,40,FE,02,02,82,42,42,82,42,42,42,82,02,02,FE,
02,01
1670 DATA FF,80,80,80,80,81,82,84,8F,80,80,80,80,F
```

```
F,80,40,FE,02,02,42,C2,42,42,42,E2,42,42,02,02,FE,
02,01
1680 DATA FF,80,80,87,84,87,80,80,80,80,87,80,80,F
F,80,40,FE,02,02,C2,02,02,82,42,42,82,02,02,02,FE,
02,01
1690 DATA FF,80,80,81,82,84,87,84,84,84,83,80,80,F
F,80,40,FE,02,02,C2,02,02,82,42,42,42,82,02,02,FE,
02,01
1700 DATA FF,80,80,87,80,80,81,82,82,82,82,80,80,F
F,80,40,FE,02,02,C2,42,82,02,02,02,02,02,02,02,FE,
02,01
1710 DATA FF,80,80,83,84,84,83,84,84,84,83,80,80,F
F,80,40,FE,02,02,82,42,42,82,42,42,42,82,02,02,FE,
02,01
1720 DATA FF,80,80,83,84,84,84,83,80,80,87,80,80,F
F,80,40,FE,02,02,82,42,42,42,C2,42,82,02,02,02,FE,
02,01
1730 DATA FF,80,80,83,84,84,85,85,86,84,83,80,80,F
F,80,40,FE,02,02,82,42,C2,42,42,42,42,82,02,02,FE,
02,01
1740 DATA FF,80,80,80,81,81,87,81,81,80,80,80,80,F
F,80,40,FE,02,02,02,02,02,C2,02,02,02,02,02,02,FE,
02,01
1750 DATA FF,80,80,80,80,80,87,80,80,80,80,80,80,F
F,80,40,FE,02,02,02,02,02,C2,02,02,02,02,02,02,FE,
02,01
1760 DATA FF,80,80,84,84,82,81,82,84,84,80,80,80,F
F,80,40,FE,02,02,42,42,82,02,82,42,42,02,02,02,FE,
02,01
1770 DATA FF,80,80,80,80,80,81,83,86,8C,80,80,80,F
F,80,40,FE,02,02,02,62,C2,82,02,02,02,02,02,02,FE,
02,01
1780 DATA FF,80,80,80,81,81,81,89,85,83,80,80,80,F
F,80,40,FE,02,02,02,F2,02,02,02,02,02,02,02,02,FE,
02,01
1790 DATA FF,80,80,80,BA,A2,B1,A1,A2,BA,80,80,80,F
F,80,40,FE,02,02,02,BA,AA,3A,22,A2,A2,02,02,02,FE,
02,01
1800 REM ** venster **
1810 DATA FF,FF,C4,C4,C4,CF,F8,C8,FF,FF,92,92,92,F
F,00,00,FF,FF,23,23,23,F3,1F,13,C8,C8,F8,C8,C8,C8,
F8,C8
1820 DATA 13,13,1F,13,13,13,1F,13,C8,F8,CF,C4,C4,C
4,FF,FF,00,00,FF,92,92,92,FF,FF,13,1F,F3,23,23,23,
FF,FF
```

```
1830 REM ** vervolg calcul.**
1840 DATA FF,80,80,80,92,AA,A3,A2,AA,92,80,80,80,F
F,80,40,FE,02,02,02,BA,AA,B2,AA,AA,AA,02,02,02,FE,
02,01
1850 DATA FF,80,80,80,AB,AA,BB,AA,AA,AB,80,80,80,F
F,80,40,FE,02,02,02,AA,2A,12,12,2A,AA,02,02,02,FE,
02,01
1860 DATA FF,80,80,80,8E,89,89,8E,88,88,80,80,80,F
F,80,40,FE,02,02,02,22,02,22,22,22,22,02,02,02,FE,
02,01
1870 DATA FF,80,80,80,80,80,87,80,87,80,80,80,80,F
F,80,40,FE,02,02,02,02,02,C2,02,C2,02,02,02,02,FE,
02,01
1880 DATA FF,80,80,92,92,92,9E,92,92,92,80,80,80,F
F,80,40,FE,02,02,22,7A,A2,72,2A,F2,22,02,02,02,FE,
02,01
1890 DATA FF,80,80,81,82,84,84,87,84,84,84,80,80,F
F,80,40,FE,02,02,02,82,42,42,C2,42,42,42,02,02,FE,
02,01
1900 DATA FF,80,80,87,82,82,83,82,82,82,87,80,80,F
F,80,40,FE,02,02,82,42,42,82,42,42,42,82,02,02,FE,
02,01
1910 DATA FF,80,80,81,82,84,84,84,84,82,81,80,80,F
F,80,40,FE,02,02,82,42,02,02,02,02,42,82,02,02,FE,
02,01
1920 DATA FF,80,80,87,82,82,82,82,82,82,87,80,80,F
F,80,40,FE,02,02,02,82,42,42,42,42,82,02,02,02,FE,
02,01
1930 DATA FF,80,80,87,84,84,87,84,84,84,87,80,80,F
F,80,40,FE,02,02,C2,02,02,82,02,02,02,C2,02,02,FE,
02,01
1940 DATA FF,80,80,87,84,84,87,84,84,84,84,80,80,F
F,80,40,FE,02,02,C2,02,02,82,02,02,02,02,02,02,FE,
02,01
1950 DATA FF,80,80,B9,A2,A2,B1,A0,A0,BB,80,80,80,F
F,80,40,FE,02,02,9A,22,22,22,A2,A2,1A,02,02,02,FE,
02,01
1960 REM ** printen **
1970 FOR I=7 TO 22:FOR J=0 TO 20:LOCATE I,J:PRINT"
▨":NEXT J,I
1980 LOCATE7,0:PRINT"┌──────────────
W┐":LOCATE7,1:PRINT"|":LOCATE7,2:PRINT"|":LOCAT
E7,3:PRINT"|":LOCATE22,1:PRINT"|":LOCATE22,2:PRI
NT"|":LOCATE22,3:PRINT"|"
1990 LOCATE7,4:PRINT"└┐▨┌────────┐▨┌
```

```
[":FOR I=5 TO 18:LOCATE8,I:PRINT"|%|":NEXT I:FOR
 I=5 TO 18:LOCATE19,I:PRINT"|%|":NEXT I
2000 LOCATE8,19:PRINT"|%└──────────┘%|":
LOCATE8,20:PRINT"└──────────┘"
2010 LOCATE11,5:PRINT"ÇéäàâêèîÄ":LOCATE11,6:PRINT"ü
åàçëïìÄ":LOCATE11,7:PRINT"ÉÆöûÿÜ£₧":LOCATE11,8:PRI
NT"æôòùÖ¢¥ƒ"
2020 LOCATE11,9:PRINT"áóñª¿¬¼«":LOCATE11,10:PRINT"
íúÑº⌐½¡»":LOCATE11,11:PRINT"ÃĩõŨĲ¾◊¶":LOCATE11,12:
PRINT"ãĩõũĳ¨¾§"
2030 LOCATE11,13:PRINT"◆┤▄Δω▄▌":LOCATE11,14:PRINT
"Ƴ ▀‡█▀":LOCATE11,15:PRINT"αΓΣµΦΩ∞€":LOCATE11,16
:PRINT"βπστθδØΠ"
2040 LOCATE11,17:PRINT"≡≥⌠÷▓▓‡Я":LOCATE11,18:P
RINT"±≤⌡≈○♂♪✱"
2050 LOCATE8,1:PRINT"▁▚▚▚▚▚▚▚▚▚▚▚▚▚▄":LOCATE8,2:PRI
NT"▔":LOCATE8,3:PRINT"▜▐▐▐▐▐▐▐▐▐▐▐▐▐▗":LOCATE21,2:P
RINT"-"
2060 B=BASE(9):RESTORE 2110
2070 FOR I=0 TO 7
2080 READ C$
2090 VPOKE B+I,VAL("&H"+C$)
2100 NEXT I
2110 DATA FF,81,81,81,81,81,81,FF
2120 RETURN
2130 REM ** einde **
2140 CLS:DEFUSR=&H69:A=USR(0)
2150 COLOR 15,4,4:END
```

Programma-6: MOVES

```
10 REM programma beweging zonder sprites / CLOAD"M
OVES" / progr.nr.6 Basis van BASIC / Ca 3000 bytes
20 CLS:SCREEN1:COLOR 10,4,4:WIDTH 32:KEY OFF:N=0
30 LOCATE 10,1:PRINT"MENU":LOCATE 0,3:PRINT"TOETS
1: MOVE-1":LOCATE0,5:PRINT"TOETS 2: MOVE-2":LOCATE
0,7:PRINT"TOETS 3: MOVE-3":LOCATE0,9:PRINT"TOETS 4
: MOVE-4":LOCATE0,11:PRINT"TOETS 5: EINDE"
40 A$=INKEY$:IF A$="" THEN 40
50 IF A$="1" THEN GOSUB 110
60 IF A$="2" THEN GOSUB 190
70 IF A$="3" THEN GOSUB 320
80 IF A$="4" THEN GOSUB 450
90 IF A$="5" THEN 820
100 CLS:PLAY"AFAF":GOTO 30
```

```
110 REM ** MOVE 1 **
120 CLS:COLOR 10,4,4:A=0
130 FOR I=1 TO 32
140 FOR D=1 TO 150:NEXT D
150 LOCATE I,12:PRINT CHR$(1);CHR$(79)
160 LOCATE I-1,12:PRINT CHR$(32)
170 NEXT I:A=A+1:IF A<3 THEN 130
180 RETURN
190 REM ** MOVE 2 **
200 CLS:COLOR 6,15,15:RESTORE 240:X=0
210 A=BASE(6):B=BASE(7)+1472
220 FOR C=184 TO 191:VPOKE A+(C/8),&H1F:NEXT C
230 FOR I=0 TO 15:READ C$:VPOKE B+I,VAL("&H"+C$):N
EXT I
240 DATA 00,00,00,00,00,02,FF,02,02,FF,02,00,00,00
,00,00
250 LOCATE 12,15:PRINT"LET OP!"
260 LOCATE 10,15:PRINT"Ĳ" '184
270 FOR D=1 TO 75:NEXT D
280 LOCATE 10,15:PRINT"ĳ" '185
290 FOR D=1 TO 75:NEXT D
300 X=X+1:IF X<15 THEN 260
310 RETURN
320 REM ** MOVE 3 **
330 CLS:COLOR 1,4,4:A=0
340 K$="...computeren is een schone zaak en geeft
het mensdom veel vermaak...        ...die informatie
 zoeken raadplegen STARK boeken... "
350 FOR I=1 TO LEN(K$)+29
360 IF I>28 THEN 390
370 LOCATE 28-I,12:PRINT MID$(K$,1,I)
380 GOTO 400
390 LOCATE 0,12:PRINT MID$(K$,I-28,28)
400 FOR J=1 TO 30:NEXT J
410 NEXT I:A=A+1
420 IF A<3 THEN 340
430 REM eventueel volgende regel t/m REM weghalen
voor INPUT en REM typen voorgaande regel
440 RETURN:REM INPUT"Andere tekst?";A$:K$=A$+" ":C
LS:GOTO 340
450 REM ** MOVE 4 **
460 CLS:COLOR 15,1,1:Y=0
470 N=N+1:IF N=1 THEN 480 ELSE 520
480 REM ** machinecode **
490 RESTORE 510
```

```
500 A!=55000!:FOR I%=0 TO 44:READ M$:POKE A!+I%,VA
L("&H"+M$):NEXT I%
510 DATA 3A,C0,DA,01,20,00,21,00,18,3C,3D,CA,EA,D6
,09,C3,E2,D6,CD,4A,00,F5,01,1F,00,23,CD,4A,00,2B,C
D,4D,00,23,0B,79,B0,C2,F1,D6,F1,CD,4D,00,C9
520 RESTORE 770
530 A=BASE(6)
540 FOR C=128 TO 143:VPOKE A+(C/8),&H41:NEXT C
550 FOR C=144 TO 159:VPOKE A+(C/8),&HA1:NEXT C
560 FOR C=160 TO 183:VPOKE A+(C/8),&H61:NEXT C
570 A=BASE(7)+1024
580 FOR I=0 TO 79:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
590 A=BASE(7)+1152
600 FOR I=0 TO 79:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
610 A=BASE(7)+1280
620 FOR I=0 TO 79:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
630 A=BASE(7)+1360
640 FOR I=0 TO 79:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
650 A=BASE(5)+136:C=128
660 FOR I=0 TO 9:VPOKE A+I,C+I:NEXT I
670 A=BASE(5)+168:C=144
680 FOR I=0 TO 9:VPOKE A+I,C+I:NEXT I
690 A=BASE(5)+200:C=160
700 FOR I=0 TO 9:VPOKE A+I,C+I:NEXT I
710 A=BASE(5)+232:C=170
720 FOR I=0 TO 9:VPOKE A+I,C+I:NEXT I
730 DEFUSR=55000!:FOR P=4 TO 8:POKE(56000!),P:A=US
R(0):NEXT P
740 FOR D=1 TO 75:NEXT D:Y=Y+1
750 IF Y<100 THEN 730
760 RETURN
770 REM ** DATA schip **
780 DATA 00,00,00,00,00,00,00,00,7F,7F,7F,30,1F,1F
,10,0F,FF,FF,FF,00,EF,EF,00,EF,FF,FF,FF,00,7F,7F,0
0,7F,FF,FF,FF,00,7E,7E,00,7F,FF,FF,FF,F8,FB,FB,F8,
FF,FF,FF,FF,00,F7,F7,00,F7,FF,FF,FF,00,EF,EF,00,EF
,FF,FF,FF,00,7F,7F,00,7F,E0,E0,E0,C0,80,80,80,00
790 DATA 00,00,0C,08,0F,00,00,00,0F,07,03,01,FF,00
,00,00,EF,EF,EF,EF,FF,00,00,00,7F,7F,7F,70,80,00,0
0,00,FF,FF,00,00,00,00,00,00,07,FF,00,00,00,00,00,
00,FF,FF,0F,00,00,00,00,00,EF,EF,EF,EF,1F,00,00,00
```

```
,7F,7E,7C,78,FF,00,00,00,00,00,03,01,FF,00,00,00
800 DATA 00,00,00,00,0F,08,0C,00,00,00,00,00,FF,00
,00,00,00,00,00,00,FF,0F,0F,0F,00,00,00,00,FF,FF,F
F,FF,00,03,0F,FF,FF,FF,FF,FF,00,FC,FF,FF,FF,FF,FF,
FF,00,00,00,F0,FF,FF,FF,FF,00,00,00,00,FF,FF,FF,FF
,00,00,00,00,FF,00,00,00,00,00,00,00,FF,01,03,00
810 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00
,00,00,0F,0F,03,01,00,00,00,00,FF,E3,E3,FF,00,00,0
0,00,FF,E3,E3,FF,01,00,00,00,FF,FC,FC,FF,F8,00,00,
00,FF,7C,7C,FF,00,00,00,00,FF,7F,7C,F8,00,00,00,00
,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
820 REM ** einde **
830 CLS:COLOR15,4,4:KEY ON:END
```

Programma-7: SLALOM

Dit is een demonstratieprogramma waarvan u wellicht het intikken achteraf minder de moeite waard zou kunnen vinden. Strikt noodzakelijk is het niet.

```
10 REM ** sprites-1 ** CLOAD"SLALOM" ** progr.nr.7
 Basis van BASIC ** Ca 1300 bytes
20 CLS:SCREEN1,1:KEY OFF:COLOR 15,1,1:K=110:R=85:S
=2:G=3:KL=1
30 A=BASE(7)+16:B=BASE(9)
40 FOR I=0 TO 7
50 C=VPEEK(A+I)
60 VPOKE B,C:B=B+1
70 NEXT I
80 Z=Z+1:IF Z<3 THEN 40:Z=0
90 A=BASE(7)+520
100 FOR I=0 TO 207
110 C=VPEEK(A+I)
120 VPOKE B+I,C
130 NEXT I:GOSUB 510
140 PUT SPRITE 0,(K,R),15:PUT SPRITE 1,(K-60,R),6:
PUT SPRITE 2,(K+60,R),4
150 FOR I=G TO 94
160 X=X+40
170 Y=RND(1)*140
180 KL=KL+1:IF KL>15 THEN KL=2
190 PUT SPRITE I,(X,Y),KL,I
200 SPRITE ON:ON SPRITE GOSUB 230
210 FOR D=1 TO 225:NEXT D
220 SWAP X,Y:GOTO 150
```

```
230 G=G+1:S=S+1:E=E+1:LOCATE 4,20:PRINT"SLALOM ";E
240 PUT SPRITE S,(0,209),0,S
250 SOUND 0,250:SOUND 1,7
260 FOR J=15 TO 0 STEP -.125
270 SOUND 6,J:SOUND 8,J
280 NEXT J
290 IF S>27 THEN 320
300 SPRITE OFF
310 RETURN
320 CLS:PUT SPRITE 1,(0,209),0:PUT SPRITE 0,(0,209
),0,0
330 LOCATE 1,22:PRINT"Afbreken met CTRL+STOP"
340 FOR I=0 TO 255 STEP 4
350 Y=Y+2:IF Y>190 THEN Y=0
360 X=X+2:IF X>255 THEN X=0
370 PUT SPRITE 21,(I,45),12
380 PUT SPRITE 14,(I+16,45),11,14
390 PUT SPRITE 17,(I+32,45),13
400 PUT SPRITE 22,(I+48,45),7
410 PUT SPRITE 28,(I,65),2,14
420 PUT SPRITE 3,(I,85),4
430 PUT SPRITE 29,(I,105),9,14
440 PUT SPRITE 2,(I-2,125),3
450 PUT SPRITE 15,(I,145),14
460 PUT SPRITE 27,(I+16,145),6,17
470 PUT SPRITE 7,(I+32,145),15
480 PUT SPRITE 26,(I+48,145),4,22
490 PUT SPRITE 30,(I+32,Y),15,0
500 NEXT I:GOTO 340
510 REM ** STEP-funktie **
520 X=X+1:IF X>255 THEN X=0
530 Y=Y+1:IF Y>191 THEN Y=0
540 PUT SPRITE 30,STEP(X,Y),11,0
550 PUT SPRITE 31,(X,Y),12,0
560 Z=Z+1:IF Z<500 THEN 510
570 PUT SPRITE 30,(X,209),0,0:PUT SPRITE 31,(X,209
),0,0
580 RETURN
```

Programma-8: ANIMAT

Kontroleert u ook hier vooral de DATA-regels. Eén foutje kan de figuren een onbedoeld vreemd aanzien geven.

```
10 REM Sprite-2 / CLOAD"ANIMAT" / progr.nr.8 Basis
```

```
 van BASIC / Ca 7000 bytes
20 CLS:SCREEN2,2:COLOR 3,1,1:KEY OFF:OPEN"GRP:"FOR
 OUTPUT AS 1
30 CLS:COLOR 3,1,1
40 LINE(0,0)-(255,191),1,BF
50 PRESET(0,16),1:PRINT#1,"TOETS 1 voor scherm 1":
PRESET(0,32),1:PRINT#1,"TOETS 2 voor scherm 2(na 5
 voor-stellingen retour)":PRESET(0,112),1:PRINT#1,
"Signaal geeft aan, dat nieuwe   toets gedrukt kan
 worden"
60 PRESET(0,56),1:PRINT#1,"ALLEEN BIJ SCHERM 1:":P
RESET(0,72),1:PRINT#1,"TOETS 3-4-5 of 6 voor anima
ties zoals op scherm aangegeven":PRESET(0,96),1:PR
INT#1,"TOETS 7 voor einde"
70 B$=INKEY$:IF B$="" THEN 70
80 IF B$=CHR$(49)THEN GOSUB 160
90 IF B$=CHR$(50)THEN 1560
100 IF B$=CHR$(51)THEN GOSUB 470
110 IF B$=CHR$(52)THEN GOSUB 670
120 IF B$=CHR$(53)THEN GOSUB 800
130 IF B$=CHR$(54)THEN GOSUB 960
140 IF B$=CHR$(55)THEN END
150 PLAY"AFAF":GOTO 70
160 CLS:LINE(0,121)-(256,121),13:LINE(116,21)-(153
,87),12,B:LINE(156,21)-(195,87),12,B:LINE(126,109)
-(142,120),11,BF:LINE(172,163)-(250,191),12,B
170 LINE(0,163)-(256,163),13:LINE(19,121)-(40,163)
,12,B:LINE(0,191)-(256,191),15:LINE(0,121)-(18,163
),14,BF
180 PSET(39,170),9:DRAW"R36D2L5D2L2D2L2D4L18U4L2U2
L2U2L5U2":PAINT(50,178),9:LINE(28,181)-(145,190),4
,BF
190 PRESET(32,182),4:PRINT#1,"eet meer gans!":PRES
ET(108,29),1:PRINT#1,"3":PRESET(198,29),1:PRINT#1,
"4":PRESET(162,171),1:PRINT#1,"5":PRESET(43,129),1
:PRINT#1,"6"
200 CIRCLE(15,15),18,15:CIRCLE(30,25),10,15:PAINT(
15,15),15:PAINT(35,22),15:PAINT(24,30),15
210 GOSUB 1220
220 REM ** modellen **
230 PUT SPRITE 0,(118,26),10,0
240 PUT SPRITE 1,(134,26),10,1
250 PUT SPRITE 2,(118,46),10,2
260 PUT SPRITE 3,(134,46),10,3
270 PUT SPRITE 4,(118,66),10,4
```

```
280 PUT SPRITE 5,(134,66),10,5
290 PUT SPRITE 6,(160,26),7,6
300 PUT SPRITE 7,(176,26),7,7
310 PUT SPRITE 8,(160,46),7,8
320 PUT SPRITE 9,(176,46),10,9
330 PUT SPRITE 10,(160,66),10,10
340 PUT SPRITE 11,(176,66),10,11
350 PUT SPRITE 12,(176,162),14,12
360 PUT SPRITE 13,(196,162),14,13
370 PUT SPRITE 15,(176,179),14,15
380 PUT SPRITE 18,(206,179),14,18
390 PUT SPRITE 16,(196,179),14,16
400 PUT SPRITE 19,(230,164),14,19
410 PUT SPRITE 14,(216,162),4,14
420 PUT SPRITE 17,(226,175),14,17
430 PUT SPRITE 20,(22,116),15,20
440 PUT SPRITE 21,(22,148),15,21
450 PUT SPRITE 22,(22,132),9,22
460 RETURN
470 REM ** beertje **
480 X=126:Y=92
490 PUT SPRITE 31,(X,Y),11,0
500 GOSUB 1190
510 PUT SPRITE 31,(X,Y),11,1
520 GOSUB 1190
530 PUT SPRITE 31,(X,Y),11,2
540 GOSUB 1190
550 PUT SPRITE 31,(X,Y),11,3
560 GOSUB 1190
570 PUT SPRITE 31,(X,Y),11,0
580 GOSUB 1190
590 PUT SPRITE 31,(X,Y),11,5
600 GOSUB 1190
610 PUT SPRITE 31,(X,Y),11,0
620 GOSUB 1190
630 PUT SPRITE 31,(X,Y),11,4
640 GOSUB 1190
650 PUT SPRITE 31,(X,Y),11,5
660 RETURN
670 REM ** eenwieler **
680 FOR X=420 TO 165 STEP -8
690 Y=88:R=104
700 PUT SPRITE 30,(X,Y),10,10
710 PUT SPRITE 29,(X,R),7,8
720 FOR D=1 TO 95:NEXT D
```

```
730 PUT SPRITE 30,(X,Y),10,11
740 PUT SPRITE 29,(X,R),7,7
750 FOR D=1 TO 95:NEXT D
760 PUT SPRITE 30,(X,Y),10,9
770 PUT SPRITE 29,(X,R),7,6
780 FOR D=1 TO 95:NEXT D
790 NEXT X:RETURN
800 REM ** olifant **
810 FOR X=472 TO 216 STEP -4
820 Y=134:R=150
830 PUT SPRITE 28,(X,Y),14,12
840 PUT SPRITE 27,(X+16,Y),14,13
850 PUT SPRITE 26,(X+16,Y),4,14
860 PUT SPRITE 25,(X,R),14,15
870 PUT SPRITE 24,(X+16,R),14,16
880 FOR D=1 TO 150:NEXT D
890 PUT SPRITE 28,(X,Y),14,17
900 PUT SPRITE 27,(X+16,Y),14,13
910 PUT SPRITE 26,(X+16,Y),4,14
920 PUT SPRITE 25,(X,R),14,18
930 PUT SPRITE 24,(X+16,R),14,19
940 FOR D=1 TO 150:NEXT D
950 NEXT X:RETURN
960 REM ** gans **
970 FOR X=40 TO 306 STEP 8
980 Y=0
990 PUT SPRITE 23,(X,Y),15,20
1000 SOUND 8,10:SOUND 7,10:SOUND 1,0:SOUND 8,15:SO
UND 0,200:SOUND 0,190:SOUND 0,180:SOUND 0,170:SOUN
D 0,180:SOUND 0,190:SOUND 0,200
1010 FOR V=10 TO 0 STEP -.125:SOUND 8,V:NEXT V
1020 PUT SPRITE 23,(X,Y),15,21
1030 FOR D=1 TO 45:NEXT D
1040 NEXT X
1050 SOUND 8,15:SOUND 13,14:SOUND 6,22:SOUND 7,33:
FOR V=15 TO 0 STEP -.125:SOUND 8,V:NEXT V
1060 SOUND 6,0:SOUND 7,62
1070 PUT SPRITE 23,(X,209),0,21
1080 FOR D=1 TO 50:NEXT D
1090 FOR Y=0 TO 160 STEP 4
1100 PUT SPRITE 0,(50,Y),9,22
1110 FOR V=10 TO 0 STEP -1
1120 FOR F=200 TO 160 STEP -125
1130 SOUND 0,F:SOUND 8,V
1140 NEXT F,V
```

```
1150 NEXT Y
1160 SOUND 0,145
1170 PUT SPRITE 0,(118,26),10,0
1180 RETURN
1190 REM ** wachtlus **
1200 FOR D=1 TO 500:NEXT D
1210 RETURN
1220 REM ** sprite-patronen **
1230 RESTORE 1280:A=BASE(9)
1240 FOR I=0 TO 735
1250 READ C$
1260 VPOKE A,VAL("&H"+C$):A=A+1
1270 NEXT I
1280 REM ** beertje **
1290 DATA 1B,37,2F,1C,3B,3B,3C,3F,3A,3B,1C,1E,0F,3
F,77,7B,EC,F6,FA,9C,6E,6E,9E,FE,AE,EE,1C,3C,F8,7E,
F7,EF
1300 DATA 1B,37,2F,1F,3F,38,3F,3F,3A,3B,1C,1E,0F,3
F,77,7B,EC,F6,FA,9C,6E,6E,9E,FE,AE,EE,1C,3C,F8,7E,
F7,EF
1310 DATA 1B,37,2F,1C,3B,3B,3C,3F,3A,3B,1C,1E,0F,3
F,77,7B,EC,F6,FA,FC,7E,0E,FE,FE,AE,EE,1C,3C,F8,7E,
F7,EF
1320 DATA 1B,37,2F,1F,3F,38,3F,3F,3A,3B,1C,1E,0F,3
F,77,7B,EC,F6,FA,FC,FE,8E,FE,FE,AE,EE,1C,3C,F8,7E,
F7,EF
1330 DATA 1B,37,2F,1C,39,3B,3C,3F,3A,3B,1C,1E,0E,3
F,77,7B,EC,F6,FA,9C,4E,6E,9E,FE,AE,EE,1C,3C,38,3E,
77,EF
1340 DATA 1B,37,2F,1C,3B,39,3C,3F,3E,3F,1C,1B,0B,3
F,77,7B,EC,F6,FA,9C,6E,4E,9E,FE,BE,FE,1C,EC,E8,7E,
F7,EF
1350 REM ** eenwieler **
1360 DATA 01,07,19,20,20,40,40,80,FF,40,40,20,20,1
9,07,01,80,E0,98,84,84,82,82,81,FF,82,82,84,84,98,
E0,80
1370 DATA 01,06,18,20,30,5C,46,81,81,42,46,24,2C,1
8,06,01,80,60,18,34,24,62,42,81,81,62,3A,0C,04,18,
60,80
1380 DATA 01,06,18,2C,24,46,42,81,81,46,5C,30,20,1
8,06,01,80,60,18,04,0C,3A,62,81,81,42,62,24,34,18,
60,80
1390 DATA 06,06,0F,06,0E,27,13,0F,01,10,1E,03,07,2
9,11,01,00,00,00,08,10,20,C0,80,C0,C0,E0,E0,E0,C0,
80,80
```

```
1400 DATA 01,01,03,01,03,01,00,01,01,01,00,01,0F,0
B,0F,19,80,80,C0,80,80,80,80,C0,C0,C0,C0,E0,E0,C0,
80,80
1410 DATA 00,00,00,08,04,02,01,00,00,01,03,03,0F,0
9,09,19,30,30,78,30,70,30,E0,F0,F0,C8,84,C2,E0,90,
88,98
1420 REM ** olifant **
1430 DATA 00,01,07,0F,0F,0D,0F,1F,37,65,4C,50,40,4
0,40,40,00,C0,60,70,70,A8,DE,FF,FF,FF,FF,FF,7F,7F,
3F,3F
1440 DATA 00,00,00,00,00,00,00,C0,F8,E7,E0,E0,E0,E
0,F8,FF,00,00,00,00,00,00,01,31,FD,3D,3F,3E,3F,3F,
FF,FF
1450 DATA 00,00,00,00,00,00,00,00,00,18,1F,1D,1A,1
D,07,00,00,00,00,00,00,00,00,00,00,C0,C0,C0,C0,C0,
00,00
1460 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,0
0,00,00,1F,0F,0F,0F,1F,3E,76,66,66,66,66,06,06,00,
00,00
1470 DATA DF,DF,DF,BF,7F,0F,00,00,01,01,01,00,00,0
0,00,00,BF,BE,DE,EE,DC,38,7C,EC,CC,8C,8C,0C,00,00,
00,00
1480 DATA 00,01,07,0F,0F,0D,0F,1F,37,25,2C,30,18,0
C,06,02,00,C0,60,70,70,A8,DE,FF,FF,FF,FF,FF,7F,7F,
3F,3F
1490 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,0
0,00,00,1F,0F,0F,0F,0F,1F,3B,73,63,63,63,63,63,00,
00,00
1500 DATA DF,DF,DF,BF,7F,0F,00,01,01,01,01,01,00,0
0,00,00,BF,BE,DE,EE,CC,2C,E8,E8,98,98,98,98,18,00,
00,00
1510 REM ** gans **
1520 DATA 00,00,00,00,00,00,00,00,9E,FF,7F,3F,3E,3
C,70,C0,00,00,00,00,00,00,1C,16,3F,70,C0,E0,F0,78,
1C,06
1530 DATA 00,00,80,C0,60,70,38,3D,BF,FF,7F,3F,1E,0
4,00,00,00,00,08,18,30,60,DC,D6,BF,70,C0,80,00,00,
00,00
1540 DATA 00,00,06,02,81,C1,E3,F7,7F,3E,00,00,01,0
0,00,00,E1,43,E7,FE,FC,F8,F0,E0,C0,C0,C0,C0,E0,A0,
E0,80
1550 RETURN
1560 REM ** tweede scherm **
1565 DEFUSR=&H69:A=USR(0)
1570 COLOR 1,4,4:A=BASE(9):W=0
```

```
1580 LINE(0,0)-(255,191),4,BF
1590 LINE(0,56)-(255,56),14:LINE(0,58)-(255,58),14
:LINE(0,61)-(255,110),12,BF
1600 FOR X=4 TO 256 STEP 24:LINE(X,56)-(X+4,60),1,
BF:NEXT X
1610 RESTORE 1640:FOR I=0 TO 95
1620 READ C$:VPOKE A+I,VAL("&H"+C$)
1630 NEXT I
1640 DATA 02,02,1F,1F,08,08,08,09,0B,0F,2F,3F,27,0
0,00,00,00,00,FC,FC,A4,E6,FF,FF,FE,CF,B7,4B,A7,94,
48,30
1650 DATA 00,00,00,00,06,0D,0B,FF,FC,CF,B7,4B,A7,9
4,48,30,00,38,38,38,38,38,38,FE,FE,CF,B7,4B,A7,94,
48,30
1660 DATA 00,42,52,24,18,28,F8,78,1D,0F,0F,07,07,0
A,0A,0A,00,00,00,00,00,00,00,78,FF,FE,FE,CC,14,24,
24,24
1670 FOR X=15 TO 290
1680 PUT SPRITE 0,(X,39),1,0
1690 PUT SPRITE 1,(X+17,39),1,1
1700 SPRITE ON:ON SPRITE GOSUB 1760
1710 NEXT X:N=N+1
1720 IF N>5 THEN 30
1730 PUT SPRITE 2,(230,39),10,2
1740 PUT SPRITE 3,(190,39),0,2
1750 A$=INKEY$:IF A$=CHR$(27)THEN 1850 ELSE 1670
1760 FOR D=1 TO 750:NEXT D
1770 PUT SPRITE 2,(0,209),0,2
1780 PUT SPRITE 3,(0,209),0,2
1790 IF Z=1 THEN 1800 ELSE 1830
1800 FOR D=1 TO 500:NEXT D
1810 PUT SPRITE 6,(0,209),0,2
1820 PUT SPRITE 7,(0,209),0,2
1830 PRESET(0,180),4:PRINT#1,"Druk ESC-toets voor
onthulling!"
1840 RETURN
1850 LINE(0,66)-(255,104),4,BF:LINE(0,100)-(255,10
0),14:LINE(0,102)-(255,102),14
1860 FOR X=4 TO 256 STEP 24:LINE(X,100)-(X+4,104),
1,BF:NEXT X
1870 Z=1:FOR X=15 TO 416
1880 PUT SPRITE 4,(X,83),1,0
1890 PUT SPRITE 5,(X+17,83),1,1
1900 SPRITE ON:ON SPRITE GOSUB 1760
1910 NEXT X
```

```
1920 PUT SPRITE 6,(190,83),10,2
1930 PUT SPRITE 7,(230,83),10,2
1940 GOTO 1670
```

Programma-9: PRENT

Let u ook hier zorgvuldig op het goede intypen van de DATA-regels a.u.b.

```
10 REM ** MSX-patronen / CLOAD"PRENT" / progr.nr.9
 Basis van BASIC / Ca 8800 bytes **
20 CLS:SCREEN 1:COLOR 1,11,11:KEY OFF:WIDTH 32:VDP
(1)=224:HX$="0123456789ABCDEF"
30 WIS=0:A=BASE(9)
40 FOR I=0 TO 640
50 VPOKE A+I,WIS
60 NEXT I
70 IF AG=5 THEN GOSUB 2680 ELSE GOSUB 1900
80 SPN=32:SN=3:SR=0:XP=184:YP=71:PX=164:PY=67:X=16
:Y=7:COL=7
90 TL=0:HR=0:VN=1:VR=1:PL=23:PM=9:LP=21:MP=9:AX=0:
BX=0:M=5
100 LOCATE18,5:PRINT USING"##";HR:LOCATE18,10:PRIN
T USING"##";HR::LOCATE21,15:PRINT" ":LOCATE28,20:P
RINT USING"##";VR:LOCATE28,20:PRINT USING"##";VN
110 LOCATE22,7:PRINT"          ":FOR I=7 TO 17:LOCA
TE21,I:PRINT" ":NEXT I
120 REM ** aanwijzing **
130 F=STICK(0)
140 IF F=1 THEN Y=Y-1
150 IF F=3 THEN X=X+1
160 IF F=5 THEN Y=Y+1
170 IF F=7 THEN X=X-1
180 PUT SPRITE 0,(X,Y),15,0
190 IF STRIG(0)THEN GOSUB 210
200 GOTO 130
210 REM ** code lezen **
220 K=INT(X/8+.25):R=INT(Y/8+.25)
230 A=BASE(5):C=VPEEK(A+(R*32+K))
240 IF K<17 THEN 250 ELSE 260
250 IF R<1 OR R>16 OR K<2 THEN 470 ELSE 270
260 IF K>17 THEN 340
270 IF C$="â" THEN 290
280 C$="â":GOTO 330
290 IF K=2 OR K=6 OR K=10 OR K=14 THEN C$="Ç"
```

```
300 IF K=3 OR K=7 OR K=11 OR K=15 THEN C$="ü"
310 IF K=4 OR K=8 OR K=12 OR K=16 THEN C$="é"
320 IF K=5 OR K=9 OR K=13 OR K=17 THEN C$="ç"
330 LOCATE K,R:PRINT C$:GOTO 470
340 IF C>143 AND C<148 THEN GOSUB 1040
350 IF C>147 AND C<152 THEN GOSUB 480
360 IF C>151 AND C<156 THEN 1300
370 IF C>155 AND C<160 THEN 630
380 IF C>183 AND C<188 THEN GOSUB 2720
390 IF C>191 AND C<196 THEN 2740
400 IF C>203 AND C<208 THEN GOSUB 1280
410 IF C=1 THEN COL=4
420 IF C=8 THEN COL=6
430 IF C=113 THEN COL=12
440 IF C=126 THEN COL=11
450 IF C=242 THEN COL=13
460 IF C=248 THEN COL=15
470 RETURN
480 REM ** achteruit **
490 IF M=0 THEN 500 ELSE 550
500 XP=XP-8:IF YP=71 THEN 530
510 IF YP>71 THEN 520
520 IF XP<192 THEN XP=216:YP=YP-8:GOTO 540
530 IF XP<192 THEN XP=192
540 BX=5:GOSUB 740:GOTO 600
550 PX=PX-16:IF PY=67 THEN 580
560 IF PY>67 THEN 570
570 IF PX<180 THEN PX=228:PY=PY-16:GOTO 590
580 IF PX<180 THEN PX=180
590 BX=5:GOSUB 840
600 HR=HR+1:LOCATE18,5:PRINT USING"##";HR
610 FOR D=1 TO 125:NEXT D
620 RETURN
630 REM ** vooruit **
640 IF M=0 THEN 650 ELSE 680
650 XP=XP+8:TL=TL+1:AX=5
660 IF XP>216 THEN YP=YP+8:XP=192
670 GOSUB 800:GOTO 710
680 PX=PX+16:TL=TL+1:AX=5
690 IF PX>228 THEN PY=PY+16:PX=180
700 GOSUB 890
710 HR=HR+1:LOCATE18,10:PRINT USING"##";HR
720 FOR D=1 TO 250:NEXT D
730 RETURN
740 REM ** pijlen/locatie **
```

```
750 PL=PL-1:IF PM=9 THEN 780
760 IF PM>9 THEN 770
770 IF PL<24 THEN PL=27:PM=PM-1:GOTO 790
780 IF PL<24 THEN PL=24
790 GOSUB 970:GOSUB 1020:GOTO 830
800 PL=PL+1
810 GOSUB 1020:GOSUB 1000
820 IF PL>27 THEN PM=PM+1:PL=24:LOCATE27,7:PRINT"
"
830 LOCATE PL-1,7:PRINT" ":LOCATE PL,7:PRINT"X":L
OCATE21,PM-1:PRINT" ":LOCATE21,PM:PRINT"/":GOTO 1
030
840 LP=LP-2:IF MP=9 THEN 870
850 IF MP>9 THEN 860
860 IF LP<23 THEN LP=29:MP=MP-2:GOTO 880
870 IF LP<23 THEN LP=23
880 GOSUB 1020:GOSUB 940:GOTO 920
890 LP=LP+2
900 GOSUB 1020:GOSUB 960
910 IF LP>29 THEN MP=MP+2:LP=23:LOCATE29,7:PRINT"
"
920 LOCATE LP-2,7:PRINT" ":LOCATE LP,7:PRINT"X":L
OCATE21,MP-2:PRINT" ":LOCATE21,MP:PRINT"/"
930 LOCATE19,6:PRINT"ǒû":LOCATE19,7:PRINT"¿ù":GOTO
 1030OK
940 VR=VR-1:IF VR<2 THEN VR=2
950 GOTO 980
960 VR=VR+1:GOTO 1010
970 VN=VN-1:IF VN<2 THEN VN=2
980 TL=TL-1:IF TL<1 THEN TL=1
990 GOTO 1010
1000 VN=VN+1
1010 LOCATE28,20:PRINT USING"##";TL:GOTO 1030
1020 LOCATE22,7:PRINT"          ":FOR I=7 TO 17:LOC
ATE21,I:PRINT" ":NEXT I
1030 RETURN
1040 REM ** 8x8 matrix **
1050 IF M=1 THEN 1060 ELSE 1080
1060 FOR I=1 TO 15:BEEP:LOCATE21,3:PRINT"¦":FOR D=
1 TO 15:NEXT D:LOCATE21,3:PRINT"{":BEEP:FOR D=1 TO
 15:NEXT D:NEXT I
1070 LOCATE21,3:PRINT" ":GOTO 1300
1080 FOR I=0 TO 26:FOR J=19 TO 22:LOCATE I,J:PRINT
"ф":NEXT J,I:LOCATE 0,19:PRINT"A}"
1090 IF BX=5 OR AX=5 THEN 1140 ELSE 1100
```

```
1100 XP=XP+8:IF XP>216 THEN YP=YP+8:XP=192
1110 PL=PL+1:GOSUB 820
1120 GOSUB 1000
1130 TL=TL+1:GOSUB 1010
1140 IF TL<25 THEN GOSUB 1220
1150 IF TL=25 THEN GOSUB 1230
1160 IF TL=26 THEN GOSUB 1240
1170 IF TL=27 THEN GOSUB 1250
1180 IF TL=28 THEN GOSUB 1260
1190 IF TL=29 THEN GOSUB 1270
1200 SN=SN+1:M=0:VDP(1)=224:GOTO 1470
1210 REM ** printroutine **
1220 LOCATE19,16:PRINT"⁘▲":LOCATE19,17:PRINT"▼▶":G
OTO 1290
1230 LOCATE19,16:PRINT" ▄":LOCATE19,17:PRINT" ▚":P
LAY"CAC":GOTO 1290
1240 LOCATE19,16:PRINT"Δω":LOCATE19,17:PRINT"‡█":P
LAY"CAC":GOTO 1290
1250 LOCATE19,16:PRINT"▄▌":LOCATE19,17:PRINT"▌▀":P
LAY"CAC":GOTO 1290
1260 LOCATE19,16:PRINT" ◆◀":LOCATE19,17:PRINT"Σ▘ ":P
LAY"CAC":GOTO 1290
1270 LOCATE19,16:PRINT"αΓ":LOCATE19,17:PRINT"βπ":P
LAY"AFAFAF"
1280 AG=5:DEFUSR=&H69:A=USR(0):GOTO 30
1290 RETURN
1300 REM ** 16x16 matrix **
1310 IF M=0 THEN 1320 ELSE 1340
1320 FOR I=1 TO 15:BEEP:LOCATE 21,0:PRINT"[":FOR D
=1 TO 15:NEXT D:BEEP:LOCATE21,0:PRINT"]":FOR D=1 T
O 15:NEXT D:NEXT I
1330 LOCATE21,0:PRINT" ":GOTO 1040
1340 FOR I=0 TO 26:FOR J=19 TO 22:LOCATE I,J:PRINT
"Φ":NEXT J,I
1350 LOCATE 0,19:PRINT"A)":LOCATE 0,20:PRINT"B)":L
OCATE 0,21:PRINT"C)":LOCATE 0,22:PRINT"D)"
1360 IF BX=5 OR AX=5 THEN 1400 ELSE 1370
1370 PX=PX+16:IF PX>228 THEN PY=PY+16:PX=180
1380 LP=LP+2:GOSUB 910
1390 TL=TL+1:GOSUB 960
1400 IF TL<17 THEN GOSUB 1220
1410 IF TL=17 THEN GOSUB 1230
1420 IF TL=18 THEN GOSUB 1240
1430 IF TL=19 THEN GOSUB 1250
1440 IF TL=20 THEN GOSUB 1260
```

```
1450 IF TL=21 THEN GOSUB 1270
1460 SR=SR+1:VDP(1)=226:M=1
1470 BX=0:AX=0:V=2:H=19:N=0:G=0:CH=0:K=1:R=1
1480 REM ** scherm-lezing **
1490 B1=0:B2=0:B3=0:B4=0:B5=0:D1=0:D2=0:D3=0:D4=0:
D5=0
1500 A=BASE(5)
1510 FOR I=1 TO 4:K=K+1
1520 IF M=1 THEN 1550
1530 IF K>9 THEN 1540 ELSE 1610
1540 K=1:K=K+1:R=R+1:IF R>8 THEN 1800 ELSE 1610
1550 IF CH=1 THEN 1590
1560 IF K>9 THEN 1570 ELSE 1610
1570 K=1:K=K+1:R=R+1:IF R>16 THEN 1580 ELSE 1610
1580 CH=1:K=9:K=K+1:R=1
1590 IF K>17 THEN 1600 ELSE 1610
1600 K=9:K=K+1:R=R+1:IF R>16 THEN 1800
1610 C=VPEEK(A+(R*32+K))
1620 IF C=176 THEN B1=0:D1=D1+B1
1630 IF C=128 THEN B2=80:D2=D2+B2
1640 IF C=129 THEN B3=40:D3=D3+B3
1650 IF C=130 THEN B4=20:D4=D4+B4
1660 IF C=135 THEN B5=10:D5=D5+B5
1670 NEXT I:N=N+1
1680 IF N>1 THEN 1730
1690 P1=D1+D2+D3+D4+D5
1700 Z1=INT((P1+1)/10)+1
1710 A$=MID$(HX$,Z1,1)
1720 GOTO 1480
1730 P2=D1+D2+D3+D4+D5
1740 Z2=INT((P2+1)/10)+1
1750 B$=MID$(HX$,Z2,1)
1760 C$=A$+B$:D$=C$+",":N=0
1770 IF V>24 THEN V=2:H=H+1
1780 LOCATE V,H:PRINT D$;:GOSUB 1800
1790 V=V+3:GOTO 1480
1800 REM ** figuur weergeven **
1810 HR=0:LOCATE18,5:PRINT USING"##";HR:LOCATE 18,
10:PRINT USING"##";HR
1820 S=BASE(9)+SPN:SPN=SPN+1
1830 VPOKE S,VAL("&H"+C$)
1840 IF M=1 THEN 1870
1850 G=G+1:IF G<8 THEN 1860 ELSE 120
1860 PUT SPRITE VN,(XP,YP),COL,SN:GOTO 1890
1870 G=G+1:IF G<32 THEN 1880 ELSE 120
```

```
1880 PUT SPRITE VR,(PX,PY),COL,SR
1890 RETURN
1900 REM ** scherm **
1910 B=BASE(6)
1920 FOR C=0 TO 7:VPOKE B+(C/8),&H14:NEXT C:FOR C=
8 TO 15:VPOKE B+(C/8),&H16:NEXT C:FOR C=18 TO 31:V
POKE B+(C/8),&H4B:NEXT C
1930 FOR C=112 TO 119:VPOKE B+(C/8),&H1C:NEXT C:FO
R C=120 TO 127:VPOKE B+(C/8),&H1B:NEXT C
1940 FOR C=240 TO 247:VPOKE B+(C/8),&H1D:NEXT C:FO
R C=248 TO 255:VPOKE B+(C/8),&H1F:NEXT C
1950 FOR C=128 TO 135:VPOKE B+(C/8),&H11:NEXT C:FO
R C=136 TO 143:VPOKE B+(C/8),&H43:NEXT C
1960 FOR C=144 TO 151:VPOKE B+(C/8),&H13:NEXT C:FO
R C=152 TO 159:VPOKE B+(C/8),&H17:NEXT C
1970 FOR C=160 TO 175:VPOKE B+(C/8),&H1B:NEXT C:FO
R C=176 TO 183:VPOKE B+(C/8),&H6B:NEXT C:FOR C=184
 TO 191:VPOKE B+(C/8),&H18:NEXT C
1980 FOR C=192 TO 199:VPOKE B+(C/8),&H1F:NEXT C:FO
R C=200 TO 207:VPOKE B+(C/8),&H17:NEXT C
1990 FOR C=208 TO 223:VPOKE B+(C/8),&H19:NEXT C:FO
R C=224 TO 231:VPOKE B+(C/8),&H16:NEXT C
2000 FOR C=232 TO 238:VPOKE B+(C/8),&HCC:NEXT C:FO
R C=40 TO 95:VPOKE B+(C/8),&H1C:NEXT C
2010 A=BASE(7)+224:RESTORE 2030
2020 FOR I=0 TO 15:READ C$:VPOKE A+I,VAL("&H"+C$):
NEXT I
2030 DATA 1C,1C,1C,1C,7F,3E,1C,08,08,0C,FE,FF,FE,0
C,08,00
2040 A=BASE(7)+1088
2050 FOR I=0 TO 7:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
2060 DATA FF,81,81,81,81,81,81,FF
2070 A=BASE(7)+1152
2080 FOR I=0 TO 223:READ C$:VPOKE A+I,VAL("&H"+C$)
:NEXT I
2090 A=BASE(7)+1408
2100 FOR I=0 TO 95:READ C$:VPOKE A+I,VAL("&H"+C$):
NEXT I
2110 A=BASE(7)+1536
2120 FOR I=0 TO 31:READ C$:VPOKE A+I,VAL("&H"+C$):
NEXT I
2130 A=BASE(7)+1632
2140 FOR I=0 TO 191:READ C$:VPOKE A+I,VAL("&H"+C$)
:NEXT I
```

```
2150 LOCATE27,19:PRINT"┌──┐":LOCATE27,20:PRINT
"|  |":LOCATE27,21:PRINT"└──┘"
2160 FOR I=0 TO 26:FOR J=18 TO 22:LOCATE I,J:PRINT
"▒":NEXT J:NEXT I
2170 FOR I=8 TO 18:FOR J=22 TO 30:LOCATE J,I:PRINT
"Ç":NEXT J,I
2180 LOCATE 2,0:PRINT"1234567812345678"
2190 FOR I=1 TO 8:D=D+1:LOCATE 0,I:PRINT D:NEXT I:
D=0:FOR I=9 TO 16:D=D+1:LOCATE 0,I:PRINT D:NEXT I
2200 LOCATE 0,4:PRINT"a":LOCATE 0,12:PRINT"b":LOCA
TE 18,4:PRINT"c":LOCATE 18,12:PRINT"d"
2210 LOCATE 0,18:PRINT"DATA-REGEL(S):":LOCATE22,7:
LOCATE1,0:PRINT"▒"
2220 FOR I=19 TO 20:FOR J=0 TO 18:LOCATE I,J:PRINT
"▒":NEXT J,I
2230 LOCATE22,0:PRINT"┌        ┐":LOCATE22,1:PRINT
"├────────┤":LOCATE22,2:PRINT"|        |":LO
CATE22,3:PRINT"|        |":LOCATE22,4:PRINT"|
  |":LOCATE22,5:PRINT"|        |":LOCATE22,6:PRIN
T"└────────┘"
2240 LOCATE19,0:PRINT"éÆ":LOCATE19,1:PRINT"æô"
2250 LOCATE19,6:PRINT"öû":LOCATE19,7:PRINT"òù"
2260 LOCATE19,3:PRINT"ÿÜ":LOCATE19,4:PRINT"ö¢"
2270 LOCATE19,8:PRINT"£₧":LOCATE19,9:PRINT"¥ƒ"
2280 LOCATE23,2:PRINT"áóÃĩñª":LOCATE23,3:PRINT"íúĨ
ĩÑº":LOCATE24,4:PRINT"õÕ¿¬":LOCATE24,5:PRINT"õÕ⌐½"
2290 LOCATE19,11:PRINT"ĲЖ":LOCATE19,12:PRINT"ĳ˜"
2300 LOCATE19,14:PRINT"▁▄":LOCATE19,15:PRINT"▗▔"
2310 REM ** DATA-regels ikoontjes **
2320 DATA FF,80,98,A4,A4,99,A4,A4,98,80,86,89,89,8
9,86,FF,FF,01,19,25,25,99,25,25,19,01,49,51,61,51,
49,FF
2330 DATA FF,F0,E0,C0,80,84,8C,9F,9F,8C,84,80,C0,E
0,F0,FF,FF,0F,07,03,01,01,01,F9,F9,01,01,01,03,07,
0F,FF
2340 DATA FF,80,A6,E8,AC,AA,AA,AA,A4,80,86,89,89,8
9,86,FF,FF,01,27,69,2D,2B,AB,2B,25,01,49,51,61,51,
49,FF
2350 DATA FF,F0,E0,C0,80,80,80,9F,9F,80,80,80,C0,E
0,F0,FF,FF,0F,07,03,01,21,31,F9,F9,31,21,01,03,07,
0F,FF
2360 DATA 01,23,27,11,09,27,64,FC,64,27,09,11,27,2
3,01,00,00,88,C8,10,20,C8,4C,7E,4C,C8,20,10,C8,88,
00,00
2370 DATA 00,36,45,26,14,64,00,75,25,25,25,25,00,7
```

```
F,7F,00,00,20,50,70,56,50,04,C4,04,84,0E,C4,00,FE,
FE,00
2380 DATA FF,81,BD,BD,BD,BD,BD,81,FF,00,00,04,3E,0
4,00,00,00,27,54,56,54,24,00,00,FF,81,81,81,81,81,
81,FF
2390 DATA 00,00,01,01,01,01,01,3F,3F,01,01,01,01,0
1,00,00,00,00,80,80,80,80,80,FC,FC,80,80,80,80,80,
00,00
2400 DATA 00,00,00,00,00,1F,1F,00,00,1F,1F,00,00,0
0,00,00,00,00,00,00,00,F8,F8,00,00,F8,F8,00,00,00,
00,00
2410 DATA FF,80,A2,A2,A2,AA,AA,B6,A2,80,86,89,89,8
9,86,FF,FF,01,9D,A1,A1,99,85,85,B9,01,49,51,61,51,
49,FF
2420 DATA FF,80,BA,A3,B2,A2,A2,BA,80,86,89,89,89,8
6,80,FF,FF,01,59,55,D5,55,55,59,01,49,51,61,51,49,
01,FF
2430 DATA FF,80,B9,A5,A5,B9,A5,A5,A5,80,86,89,89,8
9,86,FF,FF,01,DD,09,09,89,09,09,C9,01,49,51,61,51,
49,FF
2440 DATA 00,49,6A,5A,4A,4A,4A,49,00,01,02,02,83,4
2,82,01,00,8C,52,50,54,52,52,8C,00,80,40,C0,41,42,
41,80
2450 DATA 00,49,6A,5A,4A,4A,4A,49,00,03,80,C0,61,C
0,80,03,00,8C,52,50,54,52,52,8C,00,80,41,43,86,43,
41,80
2460 DATA 00,49,6A,5A,4A,4A,4A,49,00,01,82,C0,61,C
2,82,03,00,8C,52,50,54,52,52,8C,00,80,41,43,C6,03,
01,C0
2470 DATA 00,49,6A,5A,4A,4A,4A,49,00,00,81,C0,60,C
0,80,00,00,8C,52,50,54,52,52,8C,00,80,81,83,86,83,
81,80
2480 DATA FF,80,99,A5,A5,A5,A5,99,80,99,A5,A5,A5,A
5,99,FF,FF,01,C1,21,21,CF,01,0F,01,C9,29,29,C9,01,
09,FF
2490 RESTORE 2670:A=BASE(7)+8
2500 FOR I=0 TO 7:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
2510 RESTORE 2670:A=BASE(7)+64
2520 FOR I=0 TO 7:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
2530 RESTORE 2670:A=BASE(7)+1864
2540 FOR I=0 TO 7:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
2550 RESTORE 2670:A=BASE(7)+1936
```

```
2560 FOR I=0 TO 7:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
2570 RESTORE 2670:A=BASE(7)+1984
2580 FOR I=0 TO 7:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
2590 A=BASE(7)+904:RESTORE 2670
2600 FOR I=0 TO 7:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
2610 A=BASE(7)+1008:RESTORE 2670
2620 FOR I=0 TO 7:READ C$:VPOKE A+I,VAL("&H"+C$):N
EXT I
2630 A=BASE(7)+984:RESTORE 2650
2640 FOR I=0 TO 15:READ C$:VPOKE A+I,VAL("&H"+C$):
NEXT I
2650 DATA 40,FF,40,00,00,00,00,00,00,00,00,00,00,4
0,FF,40
2660 LOCATE23,0:PRINT"■°q@"≥"
2670 DATA FF,81,81,81,81,81,81,FF
2680 RESTORE 2710:A=BASE(9):FOR I=0 TO 7:READ C$:V
POKE A+I,VAL("&H"+C$):NEXT I
2690 LOCATE19,16:PRINT"⋱▲":LOCATE19,17:PRINT"▼▶"
2700 FOR I=0 TO 26:FOR J=19 TO 22:LOCATE I,J:PRINT
"⧱":NEXT J:NEXT I
2710 DATA 81,42,3C,24,24,3C,42,81
2720 FOR K=2 TO 17:FOR R=1 TO 16:LOCATE K,R:PRINT"
ê":NEXT R,K
2730 RETURN
2740 REM ** einde programma **
2750 DEFUSR=&H69:A=USR(0):CLS:COLOR 15,4,4:KEY ON
2760 LOCATE 6,12:PRINT"Succes er mee!"
2770 FOR I=1 TO 750:NEXT I
2780 CLS:END
```

Programma-10: SCHAAK

Dit programma is in het laatste hoofdstuk bij gedeelten in te typen, waarbij de diverse zaken afzonderlijke aandacht krijgen.

```
10 REM ** SCHAAKBORD/JOYSTICK/CLOAD"SCHAAK"/ progr
.nr.10 Basis van BASIC / Ca 10100 bytes
20 CLS:SCREEN1,2:WIDTH 32:COLOR 1,2,6:KEY OFF:X$=S
PACE$(9)
30 INPUT"informatie? J/N";A$:IF A$="J" THEN GOSUB
2290 ELSE 40
40 INPUT"speeltijd in uur/minuten";U1,M1
```

```
50 IF U1>9 OR M1>59 OR M1<1 THEN 40
60 CLS:GOSUB 2610:GOSUB 2780
70 REM ** start/herstart **
80 GOSUB 2930
90 X1=0:S1=0:M2=0:U2=0:X2=0:S2=0:M3=0:U3=0:ZET=0:S
P=1:N=0:XP=522:PX=74:X=16:Y=159:E=0:F=0:G=0
100 REM ** hoofdprogramma **
110 IF SP=1 THEN 120 ELSE 160
120 SZ=1:GOSUB 1100:ZET=ZET+1:LOCATE 25,19:PRINT Z
ET
130 Z=6:KL=1:GOSUB 270
140 GOSUB 1140:SZ=2
150 GOSUB 270:GOTO 200
160 SZ=3:GOSUB 1120
170 Z=6:KL=15:GOSUB 270
180 SZ=4:GOSUB 1140
190 GOSUB 270
200 IF SP=1 THEN 210 ELSE 230
210 IF Z=5 THEN GOSUB 1820
220 GOTO 1850  ' schaak contr.
230 IF Z=5 THEN GOSUB 1830
240 GOTO 1850  ' schaak contr.
250 IF SP=1 THEN SP=0 ELSE IF SP=0 THEN SP=1
260 GOTO 100
270 REM ** aanwijzing **
280 F=STICK(0)
290 IF F=1 THEN Y=Y-2
300 IF F=2 THEN X=X+2:Y=Y-2
310 IF F=3 THEN X=X+2
320 IF F=4 THEN X=X+2:Y=Y+2
330 IF F=5 THEN Y=Y+2
340 IF F=6 THEN X=X-2:Y=Y+2
350 IF F=7 THEN X=X-2
360 IF F=8 THEN X=X-2:Y=Y-2
370 PUT SPRITEZ,(X,Y),KL,Z
380 KEY(1)ON:KEY(2)ON:KEY(3)ON
390 ON KEY GOSUB 1590,1690,1730
400 GOSUB 2450 ' klok
410 IF STRIG(0) THEN GOSUB 430 ELSE 280
420 RETURN
430 REM ** schermlezing **
440 K=INT(X/8):R=INT(Y/8)+1
450 A=BASE(5):C=VPEEK(A+(R*32+K))
460 IF SZ=2 OR SZ=4 THEN 600
470 GOSUB 1360:GOSUB 830
```

```
480 REM ** verwerking **
490 IF C<152 THEN W=1:GOTO 530
500 IF C<176 THEN W=2:GOTO 530
510 IF C<200 THEN W=1:GOTO 530
520 IF C<224 THEN W=2
530 N=(C-124)/4:C2=C
540 GOSUB 2210  ' zetcontrole
550 ON N GOSUB 930,940,960,950,970,980,930,940,960
,950,970,980,930,940,960,950,970,980,930,940,960,9
50,970,980
560 GOSUB 1430  ' sprite weg
570 GOSUB 1390  ' sprite terug
580 ON W GOSUB 1000,1020
590 GOTO 820
600 IF C<152 THEN W=1:GOTO 650
610 IF C<176 THEN W=2:GOTO 650
620 IF C<200 THEN W=1:GOTO 650
630 IF C<224 THEN W=2:GOTO 650
640 IF C=>224 THEN 660
650 ON W GOSUB 1000,1020
660 A=BASE(5):C=VPEEK(A+(R*32+K))
670 IF C2<152 THEN CC=1:GOTO 710
680 IF C2<176 THEN CC=2:GOTO 710
690 IF C2<200 THEN CC=1:GOTO 710
700 IF C2<224 THEN CC=2
710 ON CC GOSUB 730,750
720 GOTO 770
730 IF C=232 THEN C=C2+24 ELSE C=C2
740 RETURN
750 IF C=224 THEN C=C2-24 ELSE C=C2
760 RETURN
770 GOSUB 830   ' registratie
780 GOSUB 990   ' slot zettekst
790 GOSUB 1040   ' stuk printen
800 GOSUB 1430  ' sprite weg
810 GOSUB 1450  ' promot. contr.
820 RETURN
830 REM ** registratie **
840 IF R=2 THEN L=8 ELSE IF R=4 THEN L=7
850 IF R=6 THEN L=6 ELSE IF R=8 THEN L=5
860 IF R=10 THEN L=4 ELSE IF R=12 THEN L=3
870 IF R=14 THEN L=2 ELSE IF R=16 THEN L=1
880 IF K=2 THEN K$="a" ELSE IF K=4 THEN K$="b"
890 IF K=6 THEN K$="c" ELSE IF K=8 THEN K$="d"
900 IF K=10 THEN K$="e" ELSE IF K=12 THEN K$="f"
```

```
910 IF K=14 THEN K$="g" ELSE IF K=16 THEN K$="h"
920 RETURN
930 LOCATE 2,20:PRINT"PION van ";K$;L:LOCATE 2,21:
PRINT"naar ":Z=0:RETURN
940 LOCATE 2,20:PRINT"TOREN van ";K$;L:LOCATE 2,21
:PRINT"naar ":Z=1:RETURN
950 LOCATE 2,20:PRINT"LOPER van ";K$;L:LOCATE 2,21
:PRINT"naar ":Z=3:RETURN
960 LOCATE 2,20:PRINT"PAARD van ";K$;L:LOCATE 2,21
:PRINT"naar ":Z=2:RETURN
970 LOCATE 2,20:PRINT"DAME van ";K$;L:LOCATE 2,21:
PRINT"naar ":Z=4:RETURN
980 LOCATE 2,20:PRINT"KONING van ";K$;L:LOCATE 2,2
1:PRINT"naar ":Z=5:RETURN
990 LOCATE 7,21:PRINT K$;L:RETURN
1000 REM ** vakje wissen **
1010 LOCATE K,R:PRINT"αα":LOCATEK,R+1:PRINT"αα":GO
TO 1030
1020 LOCATE K,R:PRINT"▓▓":LOCATE K,R+1:PRINT"▓▓"
1030 RETURN
1040 REM ** stuk printen **
1050 A$=CHR$(C):B$=CHR$(C+1):C$=CHR$(C+2):D$=CHR$(
C+3):BEEP
1060 LOCATE K,R:PRINT A$:LOCATE K,R+1:PRINT B$:LOC
ATE K+1,R:PRINT C$:LOCATE K+1,R+1:PRINT D$
1070 RETURN
1080 REM ** teksten **
1090 I$="wit zet":RR=16:KK=22:GOTO 1330
1100 I$="wit zet":RR=16:KK=22:GOTO 1290
1110 I$="zwart zet":RR=16:KK=21:GOTO 1330
1120 I$="zwart zet":RR=16:KK=21:GOTO 1290
1130 I$="rocade!":KK=22:RR=17:GOTO 1310
1140 I$="welke zet":RR=17:KK=21:GOTO 1330
1150 I$="gedaan!":KK=22:RR=16:GOTO 1310
1160 I$="remise!":KK=22:RR=15:GOTO 1290
1170 I$="geeft op!":KK=21:RR=15:GOTO 1330
1180 I$="promotie!":KK=6:RR=21:GOTO 1290
1190 I$="[tijd om\":KK=21:RR=15:GOTO 1330
1200 I$="nog eens?":KK=21:RR=16:GOTO 1330
1210 I$="toets J/N":KK=21:RR=17:GOTO 1330
1220 I$="[  WIT  \":KK=21:RR=14:GOTO 1290
1230 I$="[  WIT  \":KK=6:RR=20:GOTO 1290
1240 I$="[ ZWART \":KK=6:RR=20:GOTO 1290
1250 I$="[ ZWART \":KK=21:RR=14:GOTO 1290
1260 I$="WIT speelt":KK=5:RR=20:GOTO 1330
```

```
1270 I$="zwart SPEELT":KK=4:RR=20:GOTO 1330
1280 I$="verkeerde zet!":KK=3:RR=20:GOTO 1330
1290 LOCATE 21,14:PRINT X$
1300 LOCATE 21,15:PRINT X$
1310 LOCATE 21,16:PRINT X$
1320 LOCATE 21,17:PRINT X$
1330 LOCATE KK,RR:PRINT I$
1340 FOR I=1 TO 10:BEEP:NEXT I
1350 RETURN
1360 REM ** wis-routine **
1370 Y$=SPACE$(16):LOCATE 2,20:PRINT Y$:LOCATE 2,2
1:PRINT Y$
1380 RETURN
1390 REM ** sprite routine **
1400 IF SP=1 THEN KL=15 ELSE IF SP=0 THEN KL=1
1410 PUT SPRITE 6,(X,209),0,6
1420 PUT SPRITE Z,(X,Y),KL,Z:GOTO 1440
1430 PUT SPRITE Z,(X,209),0,Z
1440 RETURN
1450 REM ** promotie **
1460 IF R=2 OR R=16 THEN 1470 ELSE RETURN
1470 K=INT(X/8):R=INT(Y/8)+1
1480 O=BASE(5):C=VPEEK(O+(R*32+K))
1490 IF C=128 THEN 1530
1500 IF C=152 THEN 1540
1510 IF C=176 THEN 1560
1520 IF C=200 THEN 1570 ELSE RETURN
1530 C=144:GOTO 1550
1540 C=168
1550 GOSUB 1040:GOSUB 1360:GOSUB 1240:GOSUB 1180:Z
=4:GOTO 1990
1560 C=192:GOTO 1580
1570 C=216
1580 GOSUB 1040:GOSUB 1360:GOSUB 1230:GOSUB 1180:Z
=4:GOTO 1990
1590 REM ** rocade **
1600 GOSUB 1130
1610 IF SP=1 THEN 1650 ELSE 1620
1620 E=E+1:IF E=2 THEN 1640
1630 SP=1:SZ=1:GOSUB 1090:RETURN
1640 GOSUB 1150:FOR D=1 TO 75:BEEP:NEXT D:GOSUB 11
00:RETURN
1650 F=F+1:IF F=2 THEN 1670
1660 SP=0:SZ=3::GOSUB 1110:RETURN
1670 GOSUB 1150:FOR D=1 TO 75:BEEP:NEXT D:GOSUB 11
```

```
20:RETURN
1680 REM ** remise **
1690 GOSUB 1160
1700 WS=WS+.5:LOCATE 22,21:PRINT WS:ZS=ZS+.5:LOCAT
E 22,20:PRINT ZS
1710 GOTO 1760
1720 REM ** opgave/mat **
1730 IF SP=0 THEN 1740 ELSE 1750
1740 GOSUB 1250:GOSUB 1170:WS=WS+1:LOCATE 22,21:PR
INTWS:GOTO 1760
1750 GOSUB 1220:GOSUB 1170:ZS=ZS+1:LOCATE 22,20:PR
INT ZS
1760 GOSUB 1200:GOSUB 1210
1770 A$=INKEY$:IF A$="" THEN 1770
1780 IF A$=CHR$(74) THEN 70
1790 IF A$=CHR$(78) THEN 3190
1800 GOTO 1770
1810 REM ** positie koning **
1820 XP=BASE(5)+(R*32+K)-6144:RETURN
1830 PX=BASE(5)+(R*32+K)-6144:RETURN
1840 REM ** schaak situaties **
1850 S=BASE(5)+(R*32+K)-6144
1860 IF Z=0 OR Z=2 THEN 1900
1870 IF Z=3 OR Z=4 THEN 1990
1880 IF Z=1 OR Z=4 THEN 2070
1890 GOTO 250
1900 PD$="<D~é":PN$="B>"
1910 FOR I=1 TO 4
1920 PRD=ASC(MID$(PD$,I,1))
1930 IF I<3 THEN 1940 ELSE 1950
1940 NP=ASC(MID$(PN$,I,1))
1950 IF SP=1 THEN 1960 ELSE 1970
1960 IF S=PX+NP OR S=PX+PRD OR S=PX-PRD THEN 2160
ELSE 1980
1970 IF S=XP-NP OR S=XP+PRD OR S=XP-PRD THEN 2160
1980 NEXT I:GOTO 250
1990 REM ** loper/dame/toren **
2000 IF SP=1 THEN YY=PX ELSE IF SP=0 THEN YY=XP
2010 IF S>YY THEN 2040 ELSE 2020
2020 FOR I=S TO YY STEP 62:GOSUB 2100:NEXT I
2030 FOR I=S TO YY STEP 66:GOSUB 2100:NEXT I:GOTO
2060
2040 FOR I=S TO YY STEP -62:GOSUB 2100:NEXT I
2050 FOR I=S TO YY STEP -66:GOSUB 2100:NEXT I
2060 IF Z=4 THEN 2070 ELSE 250
```

```
2070 FOR I=S TO YY STEP 2:GOSUB 2100:NEXT I
2080 FOR I=S TO YY STEP -2:GOSUB 2100:NEXT I
2090 GOTO 250
2100 ZK=VPEEK(I+6144)
2110 IF ZK=224 OR ZK=232 THEN 2150 ELSE 2120
2120 IF SP=1 THEN 2130 ELSE 2140
2130 IF ZK=148 OR ZK=172 THEN 2160 ELSE 2150
2140 IF ZK=196 OR ZK=220 THEN 2160 ELSE 2150
2150 RETURN
2160 REM ** schaak **
2170 GOSUB 1360
2180 G=G+1:BEEP:FOR I=1 TO 50:NEXT I:LOCATE 2,20:P
RINT Y$:FOR I=1 TO 50:NEXT I:LOCATE 7,20:PRINT"SCH
AAK!"
2190 IF G<10 THEN 2180
2200 G=0:GOTO 250
2210 REM ** zet controle **
2220 IF SP=1 AND C<176 THEN 2240 ELSE IF SP=0 AND
C>172 THEN 2250
2230 RETURN
2240 GOSUB 1360:GOSUB 1260:SP=1:GOTO 110
2250 GOSUB 1360:GOSUB 1270:SP=0:GOTO 110
2260 REM ** verkeerde zet **
2270 REM
2280 REM
2290 REM ** informatie **
2300 CLS
2310 PRINT"      ***** SCHAAKBORD *****"
2320 LOCATE 5,1:PRINT STRING$(22,195)
2330 PRINT"1) Zetten door aanwijzing via      joys
tick"
2340 PRINT"2) Automatische promotie van       pion
 in dame"
2350 PRINT"3) Controle op verkeerde zet en    scha
ak"
2360 PRINT"4) Bepaling tijdlimiet"
2370 PRINT"5) Weergave zet, beurt, score en   tijd
"
2380 PRINT"6) Funtietoetsen:"
2390 PRINT
2400 PRINT"   TOETS 1: rocade; eerst toren-   zet,
 dan toets 1 en koningzet"
2410 PRINT"   TOETS 2: remise;J-TOETS voort   zett
ing met zelfde tijdlimiet   N-TOETS einde spel"
2420 PRINT"   TOETS 3: opgave; ook bij        scha
```

```
akmat. Via J of N-toets      voortzetting of einde"
2430 PRINT
2440 RETURN
2450 REM ** klok **
2460 IF SP=1 THEN 2470 ELSE 2550
2470 X1=X1+1:IF X1=7 THEN S1=S1+1:X1=0
2480 IF S1=60 THEN M2=M2+1:S1=0
2490 IF M2=60 THEN U2=U2+1:M2=0
2500 LOCATE 22,10:PRINT USING"##:##:##";U2,M2,S1
2510 IF U2=U1 AND M2=M1 THEN 2520 ELSE 2600
2520 IF SP=1 THEN 2530 ELSE 2540
2530 GOSUB 1220:GOSUB 1190:ZS=ZS+1:LOCATE 22,20:PR
INT ZS:GOTO 1760
2540 GOSUB 1250:GOSUB 1190:WS=WS+1:LOCATE 22,21:PR
INT WS:GOTO 1760
2550 X2=X2+1:IF X2=7 THEN S2=S2+1:X2=0
2560 IF S2=60 THEN M3=M3+1:S2=0
2570 IF M3=60 THEN U3=U3+1:M3=0
2580 LOCATE 22,9:PRINT USING"##:##:##";U3,M3,S2
2590 IF U2=U1 AND M3=M1 THEN 2520
2600 RETURN
2610 REM ** kleur/chars **
2620 LOCATE 10,10:PRINT"even geduld":LOCATE 13,12:
PRINT"a.u.b."
2630 B=BASE(6)
2640 FOR C=16 TO 31:VPOKE B+(C/8),&H12:NEXT C
2650 FOR C=128 TO 151:VPOKE B+(C/8),&H15:NEXT C:FO
R C=152 TO 175:VPOKE B+(C/8),&H14:NEXT C
2660 FOR C=176 TO 199:VPOKE B+(C/8),&HF5:NEXT C:FO
R C=200 TO 223:VPOKE B+(C/8),&HF4:NEXT C
2670 FOR C=48 TO 63:VPOKE B+(C/8),&H12:NEXT C:FOR
C=97 TO 122:VPOKE B+(C/8),&H12:NEXT C:FOR C=65 TO
95:VPOKE B+(C/8),&HF2:NEXT C
2680 FOR C=224 TO 231:VPOKE B+(C/8),&H55:NEXT C:FO
R C=232 TO 239:VPOKE B+(C/8),&H44:NEXT C
2690 FOR C=240 TO 247:VPOKE B+(C/8),&HFF:NEXT C:FO
R C=248 TO 255:VPOKE B+(C/8),&H11:NEXT C
2700 REM ** schaakstukken **
2710 A=BASE(7)+1024:G=0
2720 RESTORE 2840
2730 FOR I=0 TO 191:READ C$:VPOKE A,VAL("&H"+C$):A
=A+1:NEXT I
2740 G=G+1:IF G<4 THEN 2720
2750 G=0:A=BASE(7)+728:RESTORE 2910
2760 FOR I=0 TO 15:READ C$:VPOKE A+I,VAL("&H"+C$):
NEXT I
```

```
2770 RETURN
2780 REM ** stukken-sprites **
2790 A=BASE(9):G=0:RESTORE 2840
2800 FOR I=0 TO 191:READ C$:VPOKE A,VAL("&H"+C$):A
=A+1:NEXT I
2810 RESTORE 2900
2820 FOR I=0 TO 31:READ C$:VPOKE A+I,VAL("&H"+C$):
NEXT I
2830 RETURN
2840 DATA 00,00,00,00,00,00,01,03,07,0F,0F,07,01,0
1,03,0F,00,00,00,00,00,00,80,C0,E0,F0,F0,E0,80,80,
C0,F0 ' pion
2850 DATA 00,00,00,0D,0D,0F,0F,07,07,0F,0F,07,03,0
3,03,0F,00,00,00,B0,B0,F0,F0,E0,E0,F0,F0,E0,C0,C0,
C0,F0 ' toren
2860 DATA 00,00,01,03,07,0F,0E,04,00,03,03,01,03,0
3,03,0F,00,80,C0,E0,F0,E0,60,F0,E0,C0,C0,80,C0,C0,
C0,F0 ' paard
2870 DATA 00,01,01,03,03,07,07,0F,07,0F,07,03,03,0
3,03,0F,00,80,80,C0,C0,E0,E0,F0,E0,F0,E0,C0,C0,C0,
C0,F0 ' loper
2880 DATA 00,0A,0B,0F,0F,07,03,03,03,03,01,03,03,0
3,03,0F,00,50,D0,F0,F0,E0,C0,C0,C0,C0,80,C0,C0,C0,
C0,F0 ' dame
2890 DATA 00,01,07,07,01,07,1F,3F,3F,3F,1F,0F,07,0
3,0F,1F,00,80,E0,E0,80,E0,F8,FC,FC,FC,F8,F0,E0,C0,
F0,F8 ' koning
2900 DATA FF,80,C0,C0,E0,E0,E0,F0,F0,E0,E0,E0,C0,C
0,80,FF,FF,01,03,03,07,07,07,0F,0F,07,07,07,03,03,
01,FF  ' aanwijssprite
2910 DATA 00,04,86,FF,86,04,00,00 ' pijltje 1
2920 DATA 00,20,61,FF,61,20,00,00 ' pijltje 2
2930 REM ** scherm opmaak **
2940 LOCATE 0,0:PRINT"┌──────────────────
───────┐┌───────────┐"
2950 PRINT"| a b c d e f g h  ||ROCADE   |"
2960 PRINT"|8äàáóîÄ¿¬öûñªêè£₧8||°≡ key 1 |"
2970 PRINT"| àçíúìÄ⌐½òùÑºëï¥ƒ ||REMISE   |"
2980 PRINT"|7ÿüÇéÿüÇéÿüÇéÿüÇé7||°≡ key 2 |"
2990 PRINT"| ö¢üåö¢üåö¢üåö¢üå ||OPGAVE   |"
3000 PRINT"|6ααΦΦααΦΦααΦΦααΦΦ6||°≡ key 3 |"
3010 PRINT"| ααΦΦααΦΦααΦΦααΦΦ ||├─────────
W┤"
3020 PRINT"|5ΦΦααΦΦααΦΦααΦΦαα5||[      : 0|"
3030 PRINT"| ΦΦααΦΦααΦΦααΦΦαα ||°         |"
```

```
3040 PRINT"|4αα▣▣αα▣▣αα▣▣αα▣▣4||≡         |"
3050 PRINT"| αα▣▣αα▣▣αα▣▣αα▣▣ ||├─────────
W┤"
3060 PRINT"|3▣▣αα▣▣αα▣▣αα▣▣αα3|| TEKSTEN |"
3070 PRINT"| ▣▣αα▣▣αα▣▣αα▣▣αα ||├─────────
W┤"
3080 PRINT"|2▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓2||         |"
3090 PRINT"| ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ ||         |"
3100 PRINT"|1▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓1||         |"
3110 PRINT"| ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ ||         |"
3120 PRINT"| a b c d e f g h ||├─────────
W┤"
3130 PRINT"└┬──────────────────┬
[|ZET[      |"
3140 PRINT" |                  | |°        |"
3150 PRINT" |                  | |≡        |"
3160 PRINT" └──────────────────┘
Z──────────┘"
3170 LOCATE 22,20:PRINT ZS:LOCATE 22,21:PRINT WS:L
OCATE 22,8:PRINT USING"##:##";U1,M1:LOCATE 25,19:P
RINT ZET
3180 RETURN
3190 REM ** einde spel **
3200 SCREEN0:CLS:COLOR 15,4,4
3210 DEFUSR=&H69:A=USR(0)
3220 LOCATE 10,12:PRINT"tot ziens!"
3230 FOR D=1 TO 500:NEXT D
3240 CLS:KEY ON:END
```

Trefwoordenlijst